Niklas Tanskanen

Pilvipalvelusovelluksen päivittäminen: Case Procountor

Metropolia Ammattikorkeakoulu

insinööri (AMK)

tietotekniikka

Insinöörityö

16.10.2014

Metropolia

Tekijä Otsikko Sivumäärä Aika	Niklas Tanskanen Pilvipalvelusovelluksen päivittäminen: Case Procountor 29 sivua + 3 liitettä 16.10.2014
Tutkinto	insinööri (AMK)
Koulutusohjelma	tietotekniikka
Suuntautumisvaihtoehto	tietoverkot
Ohjaajat	yliopettaja Janne Salonen palvelupäällikkö Markus Björkbacka

Opinnäytetyön aiheena oli tutkia sähköisen taloushallinnon sovelluksen Procountorin päivittämistä Procountor International Oy:lle. Yritys työllistää noin 60 henkeä ja on osa Accountor-konsernia. Yrityksellä on ollut toimintaa vuodesta 2001 lähtien, ja nykyisin sitä käyttää yli 7000 yritystä ja 350 tilitoimistoa.

Tämä insinöörityö koostuu neljästä osasta, jossa ensimmäisenä käydään läpi Procountorin sovelluspalvelimen ominaisuuksia, toisessa palvelinten keskitettyä hallintaa, kolmannessa sovelluksen tietokannan päivittämistä ja viimeisessä osassa testataan laboratorioympäristössä työn havaintoja käytännössä.

Kirjoituksen tukena käytettiin useampaa kirjallisuuslähdettä sekä runsaasti Internet-lähteitä. Kirjallisuuslähteiden käyttöä rajoitti tiedon saatavuus, sillä kaikista tämän työn aiheista ei ollut saatavilla soveltuvaa kirjallisuutta. Työn toisessa osassa hyödynnettiin vertailevaa tutkimusta.

Työn tarkoituksena oli tutkia, kuinka Procountor-sovellus voitaisiin päivittää mahdollisimman pienellä käyttökatkolla. Samalla haluttiin etsiä ratkaisu keskitettyyn palvelinhallintaan, koska Procountorin siirtyminen Java Applet-tekniikasta Vaadin-sovelluskehykseen edellyttää useamman palvelimen käyttöönottoa.Tuloksena syntyi katsaus nykytilanteeseen ja valmis runko päivitysten automatisoinniksi.

Avainsanat	Tomcat, Capistrano, klusteri, MySQL, paraller deployment, palvelinhallinta, Rundeck, WinRM, Linux

Author	Niklas Tanskanen
Title	Upgrading Cloud Software: Case Procountor
Number of Pages	29 pages + 3 appendices
Date	16.10.2014
Degree	Bachelor of Engineering
Degree Programme	Degree Programme in Information Technology
Specialisation option	Data Networks
Instructors	Janne Salonen, Principal Lecturer Markus Björkbacka, Service Manager

Purpose of this thesis was to study and plan a upgrading process to the cloud financial management system Procountor for Procountor International Oy. Company has a 60 people workforce and is part of Accountor group. Procountor has been available since 2001 and is used by more than 7,000 SMEs and over 350 accounting offices.

Thesis consists of four parts. Studying application server Tomcat is topic of the first one, second is about centralized management of backend servers, third about updating database and final about creating upgrade plan and applying it in laboratory. Some literature was used but mainly Internet sources. Use of literature was limited because of availability, some of topics just don't have applicable literature available. Second part used comparative study as an research method.

Goals of this thesis was to find out a plan to upgrade Procountor with minor down-time and find a solution to centralized management of backend servers. Company has released a new, improved version of Procountor based on Vaadin and at the same time, multiple backend servers was installed to support a growing user base and transformation from Java Applet to Vaadin technology. A good review of current status of Procountor upgrade process and body to improved process was the main outcome of this thesis.

Keywords	Tomcat, Capistrano, Cluster, MySQL, paraller deployment, server management, Rundeck, WinRM, Linux

Sisällys

Liitteet

Liite 1 MySQL-tietokantapalvelimen Online-operaatiot

Liite 2 Tomcat-palvelinten konfiguraatiot

Liite 3 Rundeck-konfiguraatiot ja tulosteet

Lyhenteet

HTTP	Hypertext Transfer Protocol on protokolla, mitä selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon.
WAR	Web application ARchive on pakattu JAR-paketti, joka sisältää käännetyn java-ohjelmiston sekä määrittelyt sen ajamiseksi.
JAR	Java ARchive on paketoitu java-sovellus tai kirjasto, joka sisältää java-luokkia sekä niihin liittyvää metadataa kuten kuvia tai tekstiä.
TCP	Transmission Control Protocol on tietoliikenneprotokolla, jolla luodaan yhteyksiä tietokoneiden välillä.
WWW	World Wide Web on Internetissä toimiva kokoelma toisiinsa linkitettyjä hypertekstejä.
SSH	Secure Shell on salattuun tietoliikenteeseen tarkoitettu protokolla, jonka yleisin käyttötarkoitus on etäyhteys toiseen tietokoneeseen.
DSL	Domain Specific Language on syntaksi, joka on tarkoitettu tietyn ongelman ratkaisuun. Esimerkiksi CSS-tyylitiedostot on DSL-tiedostoja.
DDL	Data Definition Language on syntaksi tiedon määrittelyyn. Tietotekniikassa sitä käytetään yleisesti tietokantojen rakenteiden määrittelyyn.
DML	Data Manipulation Language on määrittelee syntaksin tiedon muokkamiseen.
CSS	Cascading Style Sheets on WWW-dokumenteille tarkoitettu tyylitiedosto.
SQL	Structuded Query Language on kyselykieli tietokantahakuihin.

DTMF	Distributed Management Task Force on organisaatio, joka kehittää järjestelmähallinnan standardeja.
SOAP	Simple Object Access Protocol on protokolla strukturoidun tiedon siirtoon.
WinRM	Windows Remote Management on Microsoftin Windowsille kehittämä tekniikka Windows-työasemien ja palvelimien etähallintaan.
RPM	Red hat Package Manager on pakettienhallintajärjestelmä Linuxille.
CLI	Command Line Input tarkoittaa komentorivipohjaista käyttöliittymää.
JVM	Java Virtual Machine on virtuaalikone, joka suorittaa Java-tavukoodia.
AJP	Apache JServ Protocol on binäärinen protokolla, joka on tarkoitettu HTTP-palvelimen ja Java-sovelluksen väliseen kommunikointiin.
MD4	Message Digest Algoritm on kryptograafinen algoritmi tarkisteen laskemiseen merkkijonolle.

1 Johdanto

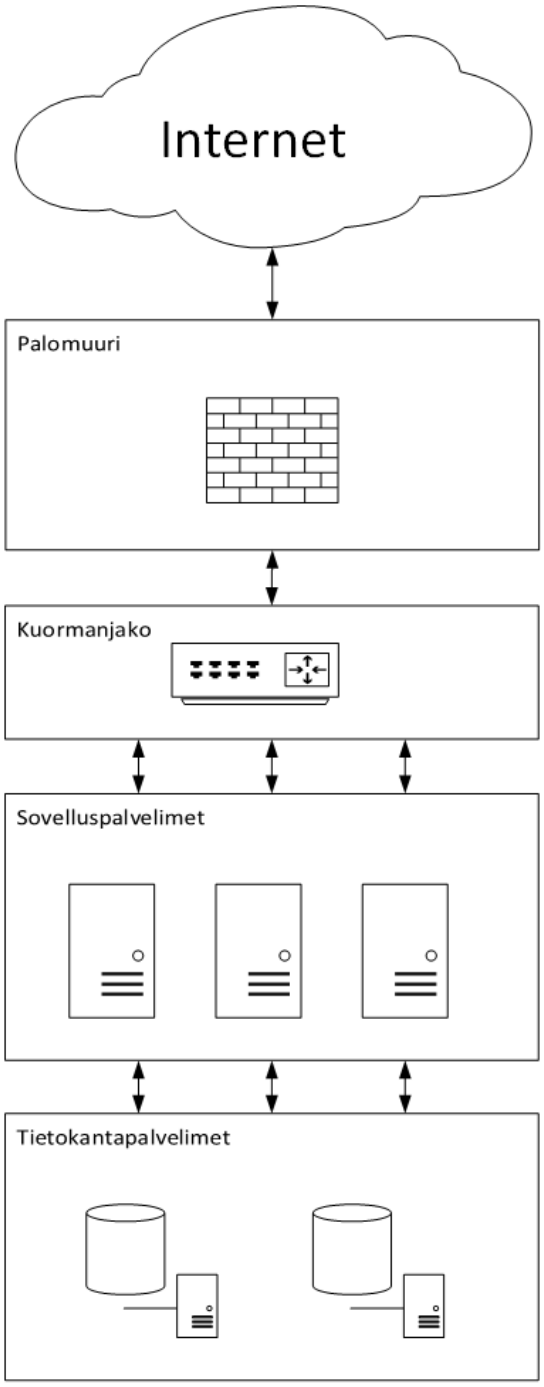
Tämän insinöörityön tavoitteena on tutkia, kuinka pilvipalvelun tuotannossa olevaa sovellusta voidaan päivittää mahdollisimman pienellä tai olemattomalla käyttökatkolla. Työ koostuu neljästä osasta, jossa ensimmäisessä käydään läpi Tomcat-sovelluspalvelimen toimintaa monipalvelinympäristössä sekä selvitetään sen ominaisuuksia työn tavoitteiden kannalta. Toisessa osassa käydään läpi päivitystyökaluja ja taustapalvelimien keskitettyä hallintaa, jotta päivitys voitaisiin ajaa keskitetysti ja hallitusti. Kolmannessa osassa tutkitaan, kuinka taustalla oleva tietokantapäivitys vaikuttaa päivityskokonaisuuteen. Työn neljännessä eli viimeisessä osassa laaditaan päivityssuunnitelma Procountorin päivittämiseksi aikaisempien havaintojen pohjalta ja toteutaan päivityssuunnitelma laboratorioympäristössä sekä arvioidaan sen onnistumista.

Procountor International Oy on suomalainen kasvuyritys, joka kehittää sähköisen taloushallinnon pilvipalvelua pohjoismaihin. Yrityksen päätuote on Java Applet -tekniikalla toteutettu Procountor-pilvipalvelu, jonka käyttäjinä ovat yritykset ja niiden tilitoimistot. Yritys on kehittänyt vanhentuneen Applet-pohjaisen ratkaisun tilalle Vaadin-sovelluskehykseen perustuvan ratkaisun.

Applet-pohjaiset sovellukset suorittavat sovelluksen logiikan vaatimaa laskentaa käyttäjän työasemalla, kun taas Vaadin-kehyksellä tehty sovellus siirtää laskennan pääosin Procountorin palvelimilla tapahtuvaksi. Procountorin tavoitteena on seuraavien vuosien aikana laajentua Pohjoismaiden markkinoille. Laajentumisen myötä käyttäjien ja datamäärän kasvu luo haasteita palvelinarkkitehtuurille, jonka tulee palvella yhä suurempia käyttäjä- ja datamääriä. Procountorin kasvu on jatkunut tasaisesti myös kotimarkkinoilla Suomessa, jossa sillä on jo yli 6000 asiakasta.

Aiempi Applet-sovelluksen palvelinarkkitehtuuri on perustunut yhteen Tomcat-palvelimeen ja taustalla toimivaan MySQL-tietokantaan. Vaadin-ratkaisun siirtyessä julkiseen pilotoin-

tiin keväällä 2014 on otettu käyttöön monipalvelinratkaisu, jonka eteen on asetettu kuormanjakaja. Yksinkertaistettu topologia on esitetty kuvassa 1. Procountorin tuotantoympäristössä on sekä Windows- että Linux-palvelimia.



Kuva 1: Pilvipalvelun topologiakuva

Kansainvälistyminen ja nopea sovelluskehitys tarkoittaa, että sovelluksesta julkaistaan useasti uusia versioita. Nykytilanteessa Procountor ei ole käytettävissä sen asiakkaille sovelluspäivityksen aikana. Päivitysrutiineihin kuuluu uuden version käyttöönotto ja tie-

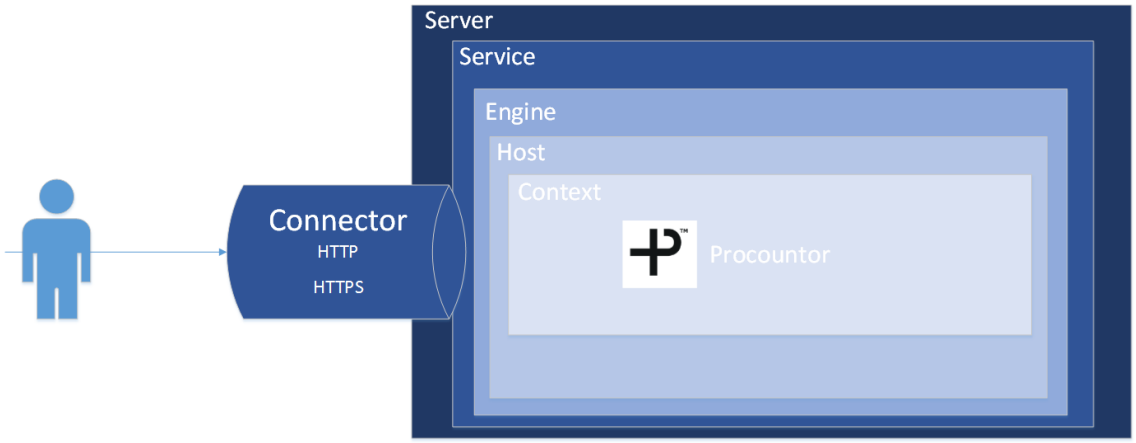
tokantapäivitykset sekä päivityksen testaaminen ennen palvelun avaamista asiakkaille. Prosessi on hitaahko, ja sen vaiheet suoritetaan käsin, jolloin riski inhimillisille virheille on suuri.

Tämä työ on tärkeä osa insinöörityötutkintoa, ja se toteutettiin itsenäisesti Procountor International Oy:lle. Työn ohjaajana toimi Metropolia Ammattikorkeakoulusta yliopettaja Janne Salonen sekä Procountor International Oy:stä palvelupäällikkö Markus Björkbacka.

2 Tomcat-sovelluspalvelin

2.1 Yleistä

Tomcat-sovelluspalvelin on Apache Foundationin kehittämä avoimen lähdekoodin palvelin, joka on tarkoitettu alustaksi Java-pohjaisten sovellusten ajoon. Sen kehittäminen alkoi vuonna 1999 ja kirjoitushetkenä tuettuna on 3 eri kehityshaaraa, 5.5.X, 6.0.X ja 7.0.X. Tomcat on vakaa ja suosittu myös kaupallisten ratkaisujen alustana. Tässä luvussa keskitytään tutkimaan Tomcatin versiohaaraa 7, sillä kyseinen versio on Procountorilla tuotantokäytössä. [1, s. 1.]



Kuva 2: Tomcatin arkkitehtuurikuva [1.]

Tomcat-palvelimen arkkitehtuuri on esitetty kuvassa 2 edellisellä sivulla. Tomcat on rakennettu pinomaisesti ja mahdollistaa tällä tavalla useiden eri sovellusten ajon samassa Tomcat-instanssissa. Sillä voi rakentaa hyvinkin monimutkaisia palveluarkkitehtuureja. Connector-elementti yhdistää käyttäjän Contextissa toimivaan sovellukseen joko HTTP(S)- tai AJP-protokollalla, joista jälkimmäistä käytetään yleensä yhdessä web-palvelimen kuten Apachen kanssa. Engine-elementti suorittaa Contextissa toimivan sovelluksen käskyjä. Tomcatin komponentit on esitetty taulukossa 1.

Taulukko 1: Tomcat-palvelimen komponentit [2.]

Komponentti	Merkitys
Catalina	Moottori, joka vastaa servlettien ajamisesta eli suorittaa Tomcatissa toimivan sovelluksen käskyjä.
Coyote	HTTP-Connector, joka vastaa HTTP-pyyntöihin ja ohjaa ne edelleen Engine-komponentille kuten Catalina.
Cluster	Monipalvelinympäristöä varten kehitetty komponentti jonka avulla voidaan suorittaa esimerkiksi kuormanjakoa. Cluster konfiguroidaan Engine-elementin alle.
Jasper	JSP-moottori Java Servlet-sivujen kääntämiseen ja näyttämiseen.

2.2 Klusteri

Tomcat-sovelluspalvelimella toimiva java-sovellus varaa itselleen resursseja käyttäjäkuorman mukaan. Kun kuorma kasvaa tarpeeksi suureksi, tulee palvelimen kapasiteettia kasvattaa, jotta kuormaa voidaan palvella luotettavasti eikä palvelimelta lopu esimerkiksi muisti. Palvelimen resursseja kuten muistia voidaan lisätä siinä toimivan käyttöjärjestelmän rajoissa. Myös java-virtuaalikone JVM, joka huolehtii sovelluksen tavukoodin suorittamisesta, asettaa tiettyjä rajoituksia sille, kuinka paljon resursseja voidaan palvelimelle lisätä.

Kun palvelimen resursseja ei voida enää kasvattaa, otetaan käyttöön toinen tai useampi palvelin. Tällöin tulee myös ottaa käyttöön kuormanjakaja, joka jakaa sovelluksen pyynnöt halutunlaisesti palvelinten välillä. Kuormanjakaja voidaan esimerkiksi asettaa ohjaamaan sovellukselle tuleva pyyntö siihen palvelimeen, missä kuormaa on vähiten. Kuormanjakaja voi olla joko sovelluspohjainen kuten Apache-palvelimelle kehitetty mod_jk tai laitteistopohjainen kuten F5 Networksin valmistama sisältökytkin Local Traffic Manager.

Useamman palvelimen joukkoa kutsutaan klusteriksi. Tomcat-palvelinten klusterointi mahdollistaa sessioiden jakamisen klusterin jäsenien kesken sekä sovelluspäivitysten jakelun klusterin jäsenille ilman käyttökatkoa. Kommunikointi palvelinten välillä tapahtuu multicast-tekniikalla. Klusterointi otetaan käyttöön tomcatin Host-elementin alle. [3.]

2.3 Sessiot

HTTP eli Hypertext Transfer Protocol on World Wide Webissä käytetty protokolla, joka toimii pyyntö/vastaus-tyyppisesti. Jokaista pyyntöä varten on avattava oma TCP-yhteys eikä edellisestä pyynnöstä ole näin saatavilla tietoja. Kun Internetin käyttö alkoi yleistymään 1990-luvulla, tuli löytää ratkaisuja, joilla voidaan pitää muistissa käyttäjän edelliset yhteydenotot palvelimelle. Esimerkiksi kirjautuminen verkkosivulla edellyttää, että palvelimella on jollain tavalla tiedossa käyttäjän aikaisemmat pyynnöt. Esimerkki asiakkaan HTTP-pyynnöstä on esitetty listauksessa 1.

```
1 GET /procountor/servlet/ProCountorVaadin HTTP/1.1
2 Host: secure.procountor.com
3 Connection: keep-alive
4 Cache-Control: max-age=0
5 Accept: text/html,application/xhtml+xml,application/xml;q
     =0.9,image/webp,*/*;q=0.8
6 User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit
     /537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 Safari
     /537.36
7 Accept-Encoding: gzip,deflate,sdch
8 Accept-Language: fi-FI,fi;q=0.8,en-US;q=0.6,en;q=0.4,sv;q
     =0.2
```

Koodiesimerkki 1: Esimerkki asiakkaan HTTP-pyynnöstä

Pyyntötilan tallentamiseen on useita ratkaisuja. Voidaan pitää esimerkiksi käyttäjän tietokoneella keksejä, jotka ovat pieniä selaimen tallentamia tekstitiedostoja. Näihin tekstitiedostoihin sovellus voi tallentaa haluamaansa tietoa, ja käyttäjän selain lähettää nämä keksit jokaisessa HTTP-pyynnössä palvelimelle. Keksit voivat sisältää esimerkiksi käyttäjän kirjautumistietoja, joista palvelin tunnistaa käyttäjän ja osaa palauttaa oikeanlaista sisältöä. Keksit voidaan myös kryptata tietoturvan parantamiseksi. Keksien haittapuolia ovat tietoturvaongelmat, jos esimerkiksi käyttäjän kirjautumistietoa tallennetaan kekseihin, voi hyökkääjä esimerkiksi varastaa nämä keksit ja tunnistautua käyttäjänä. Jos kekseissä on

paljon tietoa, HTTP-pyynnön lähettäminen palvelimelle kestää kauemmin. Myös palvelimen vasteaika kasvaa, koska keksit pitää käsitellä ennen pyyntöön vastaamista. [1, s. 95.]

Ratkaisuksi kekseistä aiheutuviin ongelmiin kehitettiin sessiot. Kun käyttäjä lähettää HTTP-pyynnön, se saa palvelimelta keksin, joka sisältää session tunnisteen (Session ID). Tämän tunnisteen avulla palvelin säilyttää tarpeellisia tietoja käyttäjän tilasta kuten esimerkiksi kirjautumisesta tai verkkokaupan ostoskorista. Sessio-tunniste voidaan pitää vaikka palvelimen muistissa tai jopa tietokannassa. Session tunniste on yksinkertainen merkkijono, mikä vähentää liikennettä palvelimen ja asiakkaan välillä verrattuna kekseihin. Esimerkki palvelimen selaimeen asettamasta keksistä, joka sisältää sessiotunnisteen JSESSIONID, on esitetty listauksessa 2 rivillä 3. [1, s. 96.]

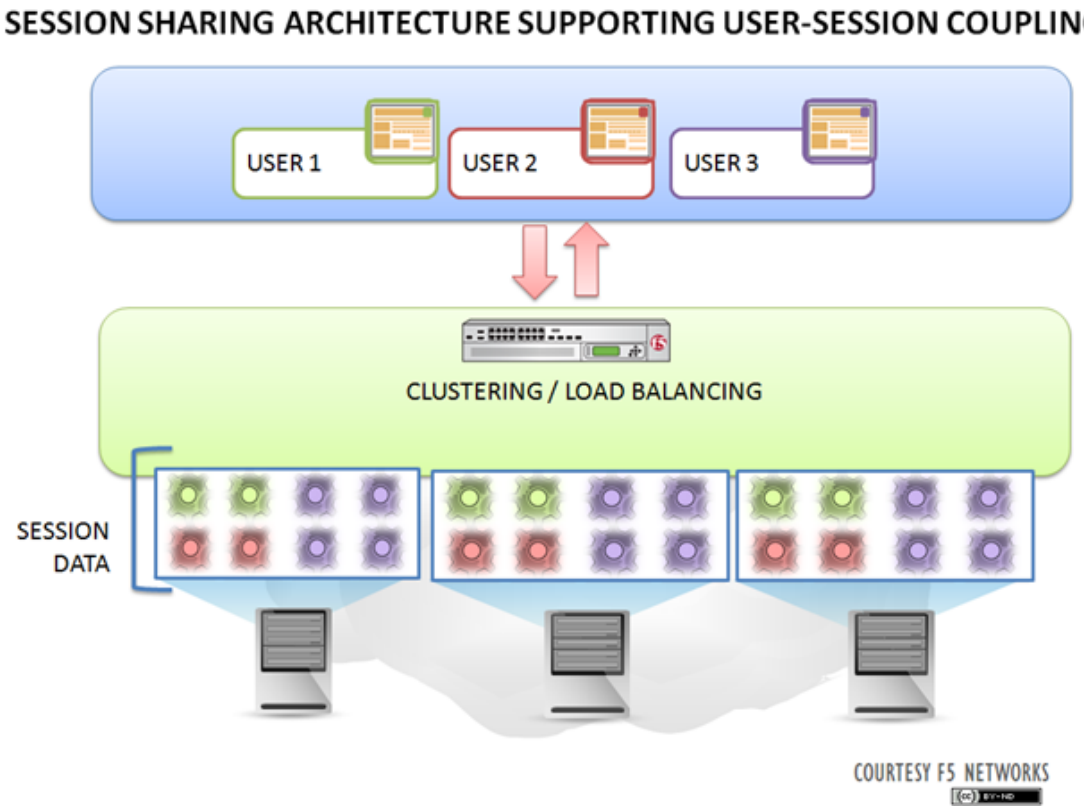
```
1 HTTP/1.1 200 OK
2 Server: Apache-Coyote/1.1
3 Set-Cookie: JSESSIONID=C01B4BCFFEE4812BF4B8F1D4762130D2;
     Path=/procountor/; Secure; HttpOnly
4 Cache-Control: no-cache
5 Pragma: no-cache
6 Expires: Thu, 01 Jan 1970 00:00:00 GMT
7 Content-Type: text/html
8 Content-Length: 1635
9 Date: Sat, 26 Jul 2014 12:38:41 GMT
```

Koodiesimerkki 2: Esimerkki palvelimen HTTP-vastauksesta

2.4 Sessiot klusterissa

Kun Tomcat-palvelimia ajetaan klusterissa, sessioiden hallinta monimutkaistuu. Mikäli klusterin jäsenillä ei ole pääsyä toistensa sessiotietoihin, tulee kuormanjakajan ohjata yhden palvelimen sessio aina samalle palvelimelle. Jos jokin klusterin jäsenistä vikaantuu, menetetään sen palvelimen sessiotiedot. Käyttäjälle tämä näkyy esimerkiksi äkillisenä uloskirjautumisena. Tomcat tarjoaa natiivisti sessiotietojen jakamiseen kahta eri menetelmää. Sessiojakamisen edellytys on, että tiedot, joita sessioihin tallennetaan, ovat serialisoitavia. Ilman serialisointia sessiotietojen jakaminen epäonnistuu, eivätkä palvelimet voi välittää sessiotietoja toisilleen. Tämä seikka on otettava huomioon sovelluskehityksen aikana. [1, s. 109.]

Ensimmäinen Tomcatin-vaihtoehto käyttää org.apache.catalina.ha.session.DeltaManager-luokkaa. DeltaManager replikoi muuttuvia sessiotietoja klusterissa käyttäen multicastia. Sen hyviä puolia on, että vain muuttuvia tietoja replikoidaan, ja kaikki palvelimet ovat tietoisia toistensa sessioista. Mikäli jokin klusterin palvelimista vikaantuu, sopii klusterin loput jäsenet vikaantuneen klusterin sessioihin vastaamisesta automaattisesti. Haittana DeltaManager-luokan käyttämiselle on lisääntynyt verkkoliikenne palvelimien välillä. Sen vuoksi se ei välttämättä sovellu suuriin klustereihin vaan klustereita on pilkottava pienempiin osiin. On myös otettava huomioon, että sessiotietojen ylläpito käyttää palvelimen resursseja kuten muistia erityisesti silloin, jos sessioihin tallennetaan paljon tietoa. [3.]



Kuva 3: Kaikilla taustapalvelimilla on sessiotiedot käytössä [4.]

Tomcatin toinen tarjoama vaihtoehto on käyttää org.apache.catalina.ha.session.BackupManager -luokkaa, jossa klusterin jäsen kopioi omat sessionsa toiselle jäsenelle talteen. BackupManagerin etuna palvelinten välinen liikenne vähentyy, ja se on siksi suositeltavampi ratkaisu isoihin klustereihin. Mikäli jokin palvelimista klusterissa vikaantuu, ohjaa kuormanjakaja käyttäjän pyynnön seuraavalle klusterin palvelimelle. Mikäli palvelin ei tunne käyttäjän sessiota, noutaa se backup-palvelimelta kyseisen session. [3.]

Sessiotiedon jakamiseen on myös muita kolmannen osapuolen kehittämiä ratkaisuja ku-

ten luokka `de.javakaffee.web.msm.MemcachedBackupSessionManager`, joka käyttää memcached-välimuistipalvelinta sessiotietojen tallentamiseen. Näitä kolmannen osapuolen ratkaisuja ei tutkittu sen tarkemmin Tomcatin luokkien osoittautuessa riittäväksi ratkaisuksi.

2.5 Sovelluksen asentaminen ja päivittäminen

WAR-paketti eli Web ARchive-paketti on valmis JAR-paketti, jolla java-ohjelmisto asennetaan Tomcat-sovelluspalvelimeen. Tomcat voidaan määrittää ottamaan WAR-paketti käyttöön ilman palvelimen uudelleenkäynnistystä. Tämä vaatii konfigurointimuutoksen server.xml:n Host-elementtiin, johon asetetaan autoDeploy-arvo. [1, s. 31-33.]

```
1 <Host appBase="webapps" autoDeploy="true" name="localhost"
      unpackWARs="true">
```

Koodiesimerkki 3: Automaattinen WAR-paketin käyttöönotto

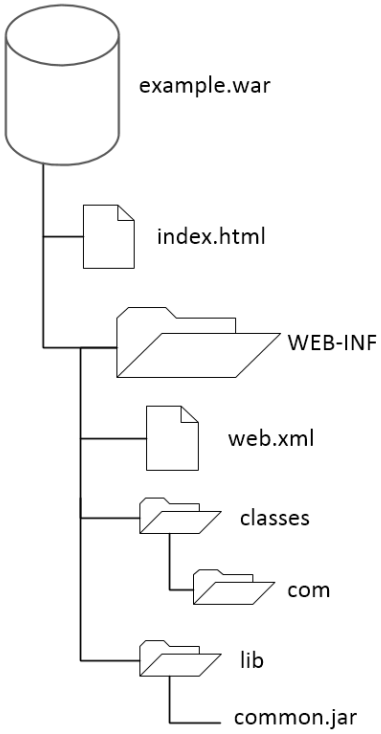
WAR-paketin rakenne on esitetty kuvassa 4 seuraavalla sivulla. Hakemiston juuressa olevat tiedostot ovat suoraan käyttäjän saatavilla lukuunottamatta WEB-INF ja META-INF hakemistoa, jossa sijaitsee sovelluksen käyttämät java-kirjastot ja luokat. [1, s. 22-23.]

WEB-INF hakemistossa sijaitsee sovelluksen web.xml-tiedosto, jossa määritellään mm.

- servletien määrittely
- servletien alustus
- sessiohallinnan asetukset
- servlet/JSP kartoitus
- MIME tyyppien kartoitus
- tietoturva-asetusten määrittely
- virhesivujen määrittely
- resurssien ja ympäristön parametrien määrittely.

Procountorin kehitysosasto julkaisee versiopäivityksen yhteydessä valmiin WAR-paketin asennettavaksi tuotantoon. Nykyisessä tilanteessa Tomcat-palvelin ajetaan hallitusti alas,

jonka jälkeen war-paketti kopioidaan Tomcat-palvelimen webapps-hakemistoon ja aiempi versio poistetaan. Sen jälkeen Tomcat-palvelin käynnistetään. Prosessi toistetaan jokaisen palvelimen kohdalla käsin.



Kuva 4: WAR-paketin rakenne

2.6 Paraller deployment

Paraller deployment on tekniikka, jossa Tomcat-sovelluspalvelimella käynnissä oleva sovellus versiopäivitetään ilman palvelimen alasajoa tai sessioiden keskeyttämistä. Tämä mahdollistaa sovelluksen päivittämisen ilman käyttökatkoa ja sujuvan siirtymisen käyttäjälle uuteen versioon. Paraller deployment toimii seuraavanlaisesti:

1. Siirretään uusi versio tomcatin webapps-hakemistoon. Tiedosto tulee nimetä sovellus##XXX.war, jossa XXX on juokseva versionumerointi.

2. Tomcat havaitsee uuden sovellusversion hakemistossa ja tunnistaa sen kuuluvan samaan contextiin, kun jo palvelimella toimiva sovellus. Jatkossa uudet sessiot luodaan tähän uuteen versioon, vanhojen sessioiden yhä jatkaessa toimintaa van-

hassa versiossa.

3. Vanha ja uusi sovellus toimivat rinnakkain, kunnes vanhassa ei ole enää voimassa olevia sessioita. Tällöin vanha versio voidaan ajaa Tomcatista alas esimerkiksi manager-työkalua käyttäen. [5.]

Paraller deployment -tekniikkaa voidaan käyttää myös klusterissa. Tällöin palvelimiin tulee konfiguroida Cluster-elementin alle Deployer-elementti listauksen 4 mukaisesti. On huomattava, että watchEnabled-parametri otetaan käyttöön vain yhdellä klusterin jäsenistä. Tältä jäseneltä paketti jaetaan muihin klusterin jäseniin. Kun watchDir-hakemistoon siirretään war-paketti, otetaan paketti käyttöön koko klusterissa välittömästi. Paketin siirtäminen tapahtuu samalla Multicast-tekniikalla kuin sessiotietojen replikointi.

```
1          <Deployer className="org.apache.catalina.ha.deploy.
               FarmWarDeployer"
2            tempDir="/application/tomcat/deployment/tempdir/"
3            deployDir="/application/tomcat/webapps/"
4            watchDir="/application/tomcat/deployment/watchdir/"
5            watchEnabled="false"/>
```

Koodiesimerkki 4: FarmWarDeployerin käyttöönotto

3 Keskitetty hallinta

3.1 Yleistä

Aiemmin käsiteltiin, kuinka Tomcat-sovelluspalvelimeen päivitetään WAR-paketti ilman käyttökatkoa käyttäjille. Sovelluksen päivittämiseen kuuluu myös muita toimenpiteitä kuten esimerkiksi Tomcatin konfiguraatiomuutoksia tai sovelluksen tarvitsemien tiedostojen tuontia palvelimille. Tarvitaan siis myös työkalu, jolla voidaan ajaa komentoja sekä siirtää tiedostoja samaan aikaan eri palvelimille. Sovellusten jakeluratkaisuja on esitelty taulukossa 2. Siitä on tarkoituksella jätetty pois sellaiset ratkaisut, jotka eivät ole enää aktiivisesti kehityksessä, jos esimerkiksi viimeinen versiojulkaisu on tapahtunut vuosia sitten.

Taulukko 2: Sovellusten jakeluratkaisuja

Ohjelma	Ohjelmointikieli	Käyttöliittymä	Arkkitehtuuri	Windows-tuki
Rundeck	Java	Web + CLI	Palvelin	Kyllä, WinRM
Fabric	Python	CLI	Palvelin	Ei
Capistrano	Ruby	CLI	Palvelin	Kyllä, WinRM

Jakeluratkaisun valintaan vaikuttaa useat seikat, joita taulukossa 2 edellisellä sivulla on esitelty. Ohjelmointikieli, jolla sovellus on ohjelmoitu auttaa käyttäjää ymmärtämään sovelluksen mahdollisuudet ja mahdollisuuden ohjelmoida työkaluun lisäosia oman ympäristön tueksi. Sovelluksen loppukäyttäjää ajatellen on hyvä tiedostaa, onko käyttöliittymä graafinen vai komentorivipohjainen. Kuten jo todettiin, taulukosta on karsittu sovellukset, joita ei aktiivisesti kehitetä. Aktiivisesti kehitetty sovellus tuo uusia ominaisuuksia ohjelmaan ja tärkeitä päivityksiä tietoturvan osalta.

Työn tavoitteiden ja topologian kannalta valinnassa suositaan jakeluratkaisuja, jotka edellyttävät mahdollisimman vähän toimenpiteitä tuotantopalvelimilta. Tällaisiä ovat esimerkiksi ratkaisut, joissa erillistä asiakasohjelmaa ei asenneta hallittaville palvelimille. [6, s. 27-28.]

Koska Procountorin palvelinympäristö sisältää myös Windows-palvelimia, on syytä tarkastella, tukeeko jakeluratkaisu Windows-palvelimia. WinRM on Microsoftin implementaatio Web Services Management -protokollasta. Web Services Management -protokolla on DMTF:n avoin standardi, jonka pohjana on SOAP. Web Services Management määrittelee palvelinten, laitteiden, sovellusten ja Web-palvelujen hallinnan. [7.]

WinRM tarjoaa palvelinten hallintaan sekä komentorivipohjaisen rajapinnan että skriptirajapinnan, joka perustuu Windows Script Host -skriptikieleen. Liikennöinti tapahtuu SOAP-kehyksillä käyttäen HTTP(S)-protokollaa. Versiosta 2.0 alkaen on ollut myös mahdollista käyttää Powershell-komentoja palvelimen hallintaan. [8.]

3.2 Rundeck

Rundeck on spin-off aikaisemmasta projektista ControlTier, jonka tavoitteet olivat samanlaiset: luoda työkalu jolla konesalien tai pilviympäristöjen keskitetty hallinta olisi helpompaa. Rundeckiä kehitetään tällä hetkellä Apache-lisenssin alaisesti SimplifyOps-yrityksen toimesta. Sen vahvuudet ovat yksinkertainen käyttöliittymä ja sen korkea käytettävyys. Rundeck on ohjelmoitu käyttäen Javaa mutta sen tunteminen ei ole välttämätöntä ohjel-

man käytön kannalta. Rundeck toteuttaa komentoja etäpalvelimilla käyttäen SSH-yhteyttä.

Rundeckillä voidaan suorittaa useita komentoja samanaikaisesti eri palvelimilla. Siihen määritellään työ ("Job"), joka sisältää yhden tai useamman askeleen ("Step") työn suorittamiseksi. Työ voidaan asettaa ajastetuksi tai suoritettavaksi web-käyttöliittymän tai API-rajapinnan kautta. Toisaalta taas yksittäisellä palvelimella voidaan suorittaa Rundeckin avulla komento aivan kuten SSH-terminaaliyhteydessä, ilman työn erillistä määrittämistä. Työn suorittaminen tapahtuu interaktiivisesti, eli käyttäjä näkee työn suorituksen tuloksen reaaliajassa. Rundeck tukee käyttöoikeuksien asettamista niin, että eri käyttäjillä on eri oikeudet ajaa ja luoda töitä rundeck-instanssiin. Jokaisen suorituksen tulosteesta jää myös loki Rundeckin tietokantaan myöhempää tarkastelua varten. [9.]

Rundeckiin on saatavilla WinRM-moduuli, joten sillä voi hallita myös Windows-palvelimia. Moduuli käyttää kommunikointiin HTTPS-protokollaa. [10.]

3.3 Fabric

Fabric on vahvasti helppona pidettyyn Python-ohjelmointikieleen nojautuva työkalu. Siinä ei ole graafista käyttöliittymää, vaan komennot suoritetaan komentoriviltä. Fabricissa määritellään tehtäviä ("Task"), jotka ovat Python-ohjelmointikielellä toteutettuja tiedostoja ("fabfile") Esimerkki tälläisestä tiedostosta on listauksessa 5. Tiedosto tulostaa tietoja palvelimen Linux-kernelistä.

```
1 from fabric.api import run
2
3 def host_type():
4     run('uname -a')
```

Koodiesimerkki 5: Esimerkki fabfile-tiedostosta

On huomioitava, että Fabricin käyttö edellyttää Python-kielen tuntemusta, itse asiassa Fabric onkin Python-kirjasto, mikä mahdollistaa helpon laajentamisen ja integroinnin muihin sovelluksiin. Fabric tarjoaa valmiit funktiot tiedostojen lataamiseen palvelimelle tai palvelimelta, käyttäjän syötteen kysymiseen ja suorituksen pysäyttämiseen.

Kun tehtävä on määritelty fabfilessä, se voidaan ajaa etäpalvelimille listauksen 6 mukaan. [11.]

```
1 $ fab -H localhost,linuxbox host_type
2 [localhost] run: uname -s
3 [localhost] out: Darwin
4 [linuxbox] run: uname -s
5 [linuxbox] out: Linux
6
7 Done.
8 Disconnecting from localhost... done.
9 Disconnecting from linuxbox... done.
```

Koodiesimerkki 6: Esimerkki Fabricin käytöstä

3.4 Capistrano

Capistrano on Ruby-ohjelmointikielellä toteutettu ratkaisu, joka luotiin helpottamaan websovellusten jakelua useamman palvelimen ympäristöön kuten pilvipalvelimiin. Capistrano soveltuu parhaiten Ruby on Rails -sovellusten jakeluun mutta soveltuu myös muuhun käyttöön. Ruby-ohjelmointikieltä on syytä osata Capistranoa käyttäessä. Capistrano onkin suhteellisen samanlainen aikaisemmin esitellyn Fabricin kanssa, erona se, että Capistranoa käytetään Rubyllä ja siinä on hieman enemmän ominaisuuksia kuten tulosteen automaattinen muotoilu esimerkiksi HTML:ksi. Capistrano sisältää myös hieman työkaluja tietokannan päivitykseen. [12.]

Esimerkki DSL-skriptistä, jota Capistrano käyttää on esitelty listauksessa 7. Esimerkki tulostaa ajan, jonka palvelin on ollut päällä.

```
1 role :demo, %w{example.com example.org example.net}
2 task :uptime do
3   on roles(:demo), in: :parallel do |host|
4     uptime = capture(:uptime)
5     puts "#{host.hostname} reports: #{uptime}"
6   end
7 end
```

Koodiesimerkki 7: Esimerkki Capistrano taskista

3.5 Valinta

Ratkaisujen tutkimisen jälkeen tehtiin valinta, mitä työkalua tutkittaisiin tarkemmin laboratorioympäristössä Procountorin käyttöä ajatellen. Valittaessa annettiin paljon painoarvoa helppoudelle. Valittavan ratkaisun on oltava sellainen, jota on helppo käyttää ja ymmärtää. Myös laajennettavuudelle Windows-palvelimiin kiinnitettiin huomiota sekä myöhempään laajennettavuuteen konfigurointihallinnan kuten Puppetin tai Chefin kanssa. Vaihtoehtojen tarkempi tutkiskelu osoitti, että Rundeck soveltuu Procountorin nykyiseen tilanteeseen parhaiten: sen käyttöliittymä on ylivertaisen helppo käyttää, eikä se vaadi tuntemusta mistään ohjelmointikielestä ja sillä on vahva käyttäjäkunta takanaan. Se myös integroituu helposti muihin järjestelmiin sekä kykenee hallitsemaan myös Windows-palvelimia.

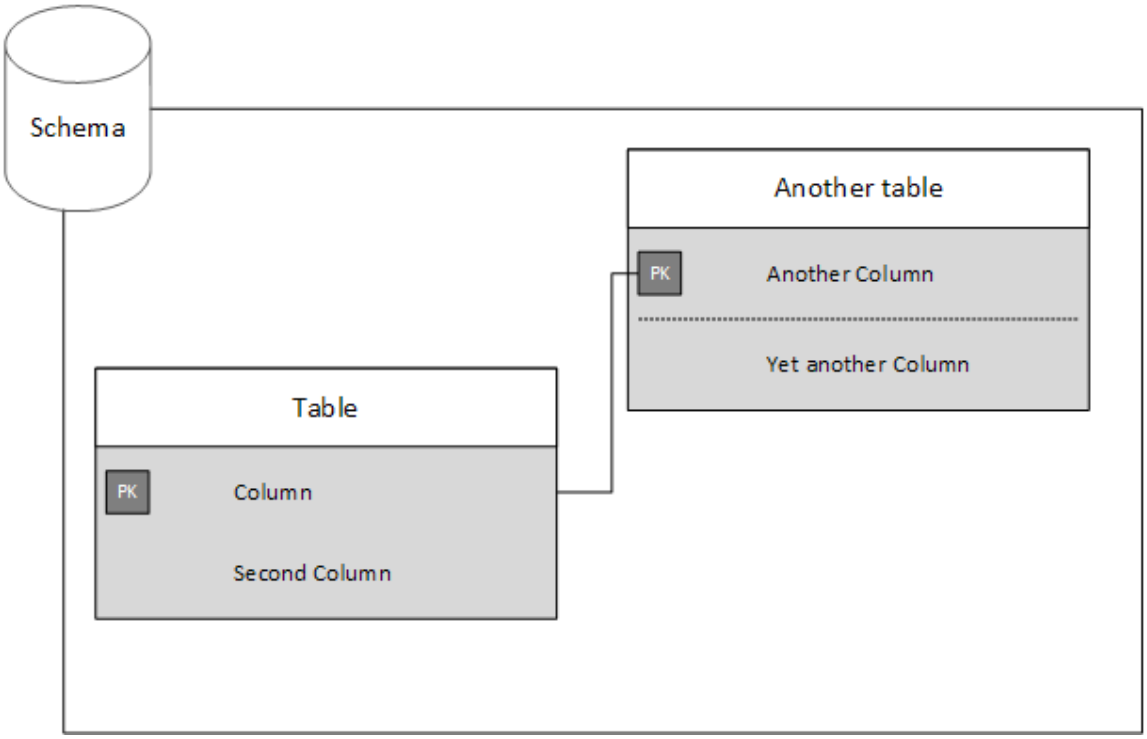
4 Tietokannan päivittäminen

4.1 Yleistä

MySQL on hyvin suosittu avoimen lähdekoodin tietokantapalvelinohjelmisto. Sen kehittäminen alkoi avoimen lähdekoodin projektina vuonna 1995. Nykyään MySQL:n omistaa yhdysvaltalainen Oracle. MySQL:n arkkitehtuurissa keskeinen osa on tietokantamoottori, joka vastaa tietokannan toiminnoista. Palvelimella voi käyttää useita eri moottoreita kuten MyISAMia tai InnoDB:tä joista jälkimmäinen on käyttäjäkunnassa suosituin. Procountorin käytössä on InnoDB-moottori. [13, s. 12-13.]

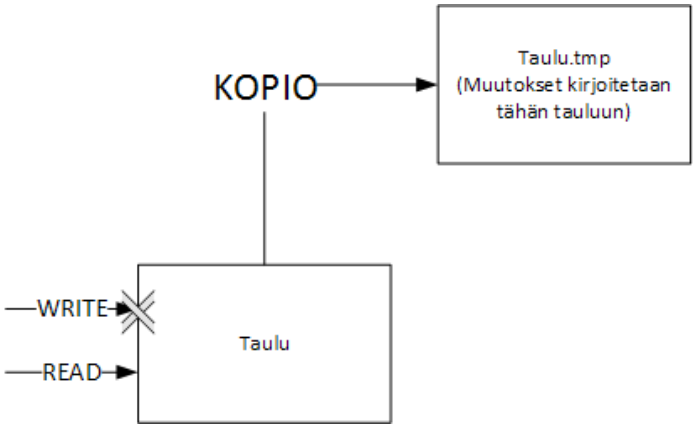
MySQL koostuu tietokannoista ("schema") ja niiden sisällä olevista tauluista ("table") jotka sisältävät sitä käyttävän sovelluksen tuottamat tiedot. Taulujen kentät ("column") ovat erilaisia tietotyyppejä kuten kokonaislukumuuttujia tai merkkijonoja.Esimerkki relaatiotietokannasta on esitetty kuvassa 5 seuraavalla sivulla. DML eli Data Manipulation Language on kieli, joka kuvaa tietokannan tietojen muokkaamista.MySQL:n tietokannan tauluja luetaan ja muokataan standardin mukaisella SQL-kielellä. Huolimatta standardista jokaisella tietokantapalvelimella on SQL:sta oma implementaatio ja eri alustojen väliset komennot

voivat vaihdella paljonkin. [13.]



Kuva 5: Relaatiotietokanta

Tietokannan taulujen rakenteen päivittäminen on perinteisesti ollut mahdotonta ilman taulun kirjoitusoperaatioiden lukitsemista. Tämä ei ole kovin iso ongelma, kun taulut ovat pieniä, mutta taulukoon kasvessa suureksi menee muutosoperaatioihin kauemmin. Koska kirjoitusoperaatiot estetään, ei tietokantaa käyttävää sovellusta voi käyttää, kun tietokantaa päivitetään. Tämä näkyy sovelluksen käyttäjille käyttökatkona. Esimerkki on esitetty kuvassa 6 seuraavalla sivulla, jossa MySQL-palvelin estää kirjoitusoperaatiot tauluun, luo siitä väliaikaisen kopion ja tekee muutokset kopioituun tauluun. Kun muutokset on saatu valmiiksi, alkuperäinen taulu poistetaan, ja taulu, johon muutokset tehtiin, nimetään uudelleen alkuperäisen taulun nimellä. [14.]



Kuva 6: MySQL-tietokannan päivitysprosessi

4.2 Päivitystapoja

4.2.1 Slave-palvelimen päivitys

Master- ja slave -replikointia käytetään tietojen varmuuskopiointiin ja/tai jakamaan tietokantakuormaa muille palvelimille. Master-palvelille tehdyt muutokset replikoidaan ylläpitäjän määrittelemän aikataulun mukaisesti slave-palvelimelle. Mikäli master-palvelin vikaantuu, voidaan slave-palvelimen rooli muuttaa master-palvelimeksi ja ottaa nopeasti käyttöön ilman muuta varmuuskopioiden palautusta. Kun master-palvelimen vikatilanne saadaan selvitettyä, voidaan se muuttaa slave-palvelimeksi eli vaihtaa palvelinten rooleja keskenään. [14.]

Tuotantosovelluksen päivittämisessä voidaan käyttää tätä roolien vaihtoa. Ajetaan slave-palvelimelle sovelluksen päivityksen yhteydessä tarvittava päivitys. Kun slave-palvelimen tietokantapäivitys on valmis, vaihdetaan palvelinten roolit niin, että slave-palvelimesta tulee master-palvelin ja master-palvelimesta slave. Samalla päivitetään tuotantosovellus uuteen versioon. Aiemmalle master-palvelimella, joka nyt toimii slavena replikoidaan päivityksen aiheuttamat muutokset. [14.]

Tämäkin ratkaisu aiheuttaa pienen käyttökatkon tietokannan käyttöön, kun sovellukselle täytyy määritellä uusi käytettävä tietokantapalvelin. Se ei tosin kestä niin kauaa kuin aiemmin mainittu ALTER TABLE -komento mutta näkyy pienenä käyttökatkona käyttäjil-

le. [14.]

4.2.2 Päivittäminen ilman taulun lukitsemista

MySQL:n versiopäivitys 5.6, joka julkaistiin vuonna 2011, toi Online DDL -ominaisuuden, missä parannettiin tietokannan päivitysmahdollisuuksia. Aikaisemmin osa `ALTER TABLE` -operaatioista vaati taulun kirjoituslukituksen ja rinnakkaisen taulun muodostuksen ja esti DML-komentojen käytön operaation aikana. Versiosta 5.6 alkaen ilman taulun lukitsemista onnistuu esimerkiksi

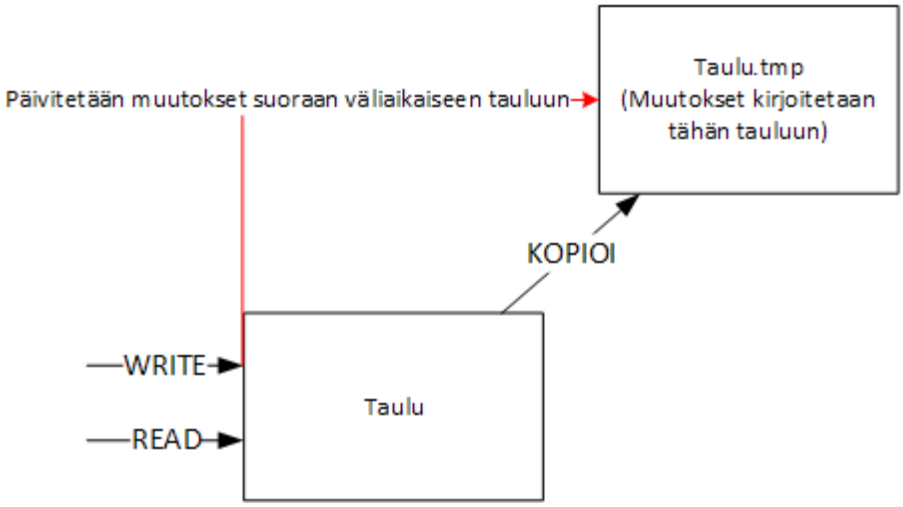
- auto-increment arvon muuttaminen taulun kentälle
- kentän nimen muuttaminen. (jos tietotyyppi säilyy samana)

Täydellinen taulukko siitä, mitä operaatioita voidaan suorittaa ilman taulun kopiointia ja kirjoitusoperaation estämistä, löytyy liitteestä 1.

Online DDL toimii niin, että taulun päivityksen aikana tauluun kohdistuvat päivitykset ajetaan vasta, kun päivitys on saatu valmiiksi. Taulusta ei tehdä kopioita, vaan muutokset tehdään suoraan muokattavana olevaan tauluun. [15.]

Vastaava tietokantapäivitys on mahdollista tehdä myös MySQL:n aikaisemmilla versioilla. Tähän löytyy useita työkaluja kuten Facebookin kehittämä Online Schema Change. Online Schema Change on PHP-skriptikielellä toteutettu työkalu, mikä aloittaa tekemällä muutettavasta taulusta kopion ja tekemällä muutokset siihen. Sen sijaan, että alkuperäinen taulu lukittaisiin, tehdään MySQL-tietokantaan väliaikainen TRIGGER-komento, mikä kirjoittaa alkuperäisen tauluun kohdistuvat muutokset suoraan kopioitavaan tauluun kopioinnin aikana. Esimerkki on esitetty kuvassa 7 seuraavalla sivulla. Tällöin tauluun kirjoitusta ei tarvitse lukita. Väliaikaisen taulun kopiointi tapahtuu pienissä erissä, jolloin tietokantapalvelin suorituskyky ei vaarannu ja sovelluksen toiminta hidastu tietokannan hidastumisen vuoksi. [16.]

Toinen vastaava työkalu on yhdysvaltalaisen Perconan kehittämä `pt-online-schema-change`, joka käyttää vastaavanlaista strategiaa taulun rakenteen muuttamiseksi. Siinä, missä Facebookin kehittämä ratkaisu on tehty PHP:llä, on Perconan ratkaisu tehty Perlillä, joka on huomattavasti laajemmassa käytössä ja löytyy useista palvelinympäristöistä valmiina.

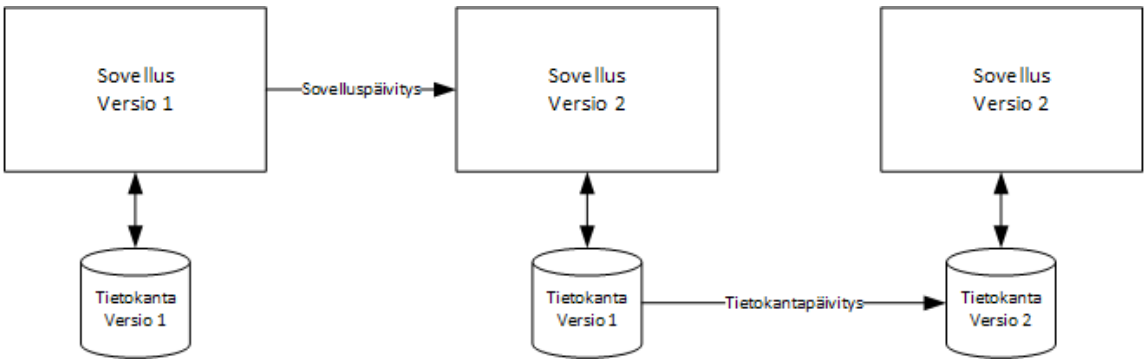


Kuva 7: MySQL-tietokannan päivitysprosessi

4.3 Yhteenveto

Tietokannan päivittämisen tarkempi tutkiminen osoitti, että tämän työn puitteissa toteutettavaa suunnitelmaa tietokannan päivittämiseksi ei ole mahdollista tehdä. Lähestymistavasta riippuen tietokantapäivitysi ilman käyttökatkoa vaatii muutoksia palvelin- tai sovellusarkkitehtuuriin. Tutkiminen oli kuitenkin hyödyllistä ja antoi näkemyksiä tulevaisuuden arkkitehtuurimuutoksiin.

Päivityksiä helpottaisi, jos sovellus olisi eteenpäin yhteensopiva niin, että tietokannan voisi päivittää ennen sovellusta eli sovellus olisi tietoinen tietokantaversiosta ja pystyisi sopeuttamaan tietokannan käyttöä sen version mukaan. Esimerkki tällaisestä päivitysprosessista on esitetty kuvassa 8 seuraavalla sivulla



Kuva 8: Sovelluksen päivitysjärjestys

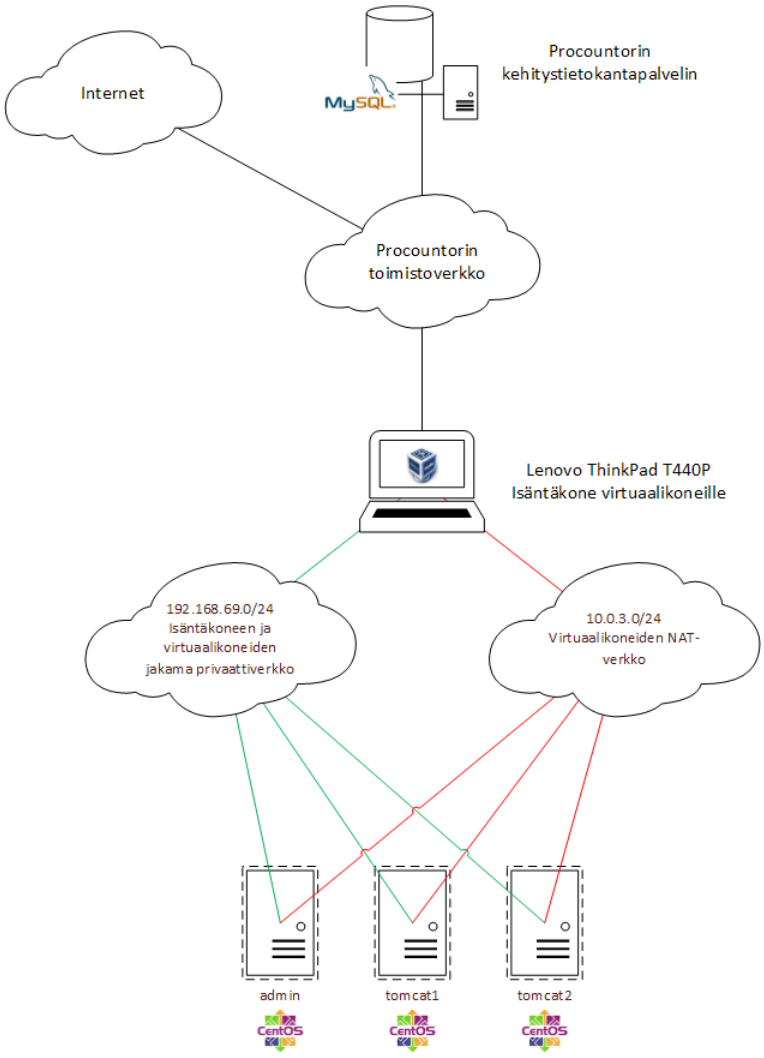
5 Käyttöönotto laboratorioympäristössä

Tässä luvussa tehdään Rundeckin käyttöönotto laboratorioympäristössä sekä tutkitaan Procountorin käyttäytymistä Tomcatin klusteroidussa ympäristössä. Käyttöönottoa varten asennettiin käytössä olevalle Lenovo ThinkPad T440p -työasemalle Oracle Virtualbox -virtualisointialusta. Siihen asennettiin kolme virtuaalikonetta, joiden roolit on esitetty taulukossa 3 seuraavalla sivulla.

Kaksi Tomcat-palvelinta asennettiin mahdollistamaan palvelinten klusteroinnin. Rundeckin asennusta varten luotiin erillinen admin-palvelin sekä rundeckiä että palvelinten yhteistä levyjakoa varten.

Palvelinten käyttöjärjestelmäksi valittiin CentOS 6.5, koska se vastaa parhaiten Procountorin tuotantoympäristön käyttöjärjestelmää. Virtuaalikoneille asennettiin kaksi verkkokorttia: toinen Host-only tilaan isäntäkoneen verkkoa varten ja toinen NAT-tilaan, jotta virtuaalikoneilla olisi pääsy Internetiin ja Procountorin kehitystietokantapalvelimille. Virtuaalikoneiden topologia on esitetty kuvassa 9 seuraavalla sivulla.

Virtuaalikoneiden Host Only -verkon IP-avaruudeksi asetettiin 192.168.69.0/24 ja NAT-verkolle 10.0.3.0/24. VirtualBox sisältää valmiiksi NAT-ominaisuuden ja DHCP-palvelimen NAT-verkkoa varten. Host Only-verkkokorteille konfiguroitiin staatiset IP-osoitteet.



Kuva 9: Laboratorioympäristön topologia

Taulukko 3: Virtuaaliympäristön palvelimet

Palvelin	Rooli	IP-osoite
tomcat1.procountor.local	Tomcat-palvelin osana klusteria	192.168.69.100
tomcat2.procountor.local	Tomcat-palvelin osana klusteria	192.168.69.101
admin.procountor.local	Rundeck-palvelin ja levyjako	192.168.69.200

Tomcat-palvelimille luotiin samanlainen hakemistorakenne, jotta palvelinten hallinta olisi mahdollisimman helppoa. Hakemistorakenne on esitetty taulukossa 4 seuraavalla sivulla Lisäksi admin-palvelimelle luotiin NFS:llä jaettu hakemisto `/media`, jonne kopioitiin sovelluksen tarvitsemia jaettuja tiedostoja kuten laskupohjia. Sovelluksen testipalvelimelta kopioitiin vielä sovelluksen konfiguraatiotiedostot `/lab/procountor/`-hakemistoon.

Tomcatin asennus tomcat-palvelimille tapahtui noutamalla uusin versio Tomcatin 7-versiosta

Apachen-kotisivuilta. Tomcat purettiin hakemistoon `/lab/apache-tomcat-7.0.55` Asennuksen jälkeen haettiin Jenkins-kehitysjärjestelmästä Procountor-sovelluksesta uusin versio. Kehitysversio asetettiin hakemistoon /lab/apache-tomcat-7.0.55/webapps/procountor.war. Klusterointia varten Tomcatin konfiguratioihin tehtiin muutoksia. Myös Tomcatin manager-liittymä haluttiin ottaa käyttöön sessiotietojen ja palvelimella toimiven sovellusten tarkastelua varten. Tarkemmat konfiguraatiot Tomcat-palvelimilta on esitetty liitteessä 2.

Taulukko 4: Tomcat-palvelinten hakemistorakenne /lab/

Hakemisto	Merkitys
apache-tomcat-7.0.55	Tomcat-palvelimen asennushakemisto
procountor	Procountor-sovelluksen hakemisto
shared	Procountor-sovelluksen jaetut tiedostot

Koska laboratioympäristössä ei ollut erillistä DNS-palvelinta, konfiguroitiin jokaiselle virtuaaliympäristön palvelimelle `/etc/hosts`-tiedostoon virtuaalipalvelimien isäntänimet ja IP-osoitteet. Admin-palvelimelle asennettiin Rundeck noutamalla siitä uusin versio sovelluksen kotisivuilta. Koska rundeckille oli saatavilla valmis RPM-paketti, oli asennus suoraviivaista ja helppoa. Asennus on esitetty listauksessa 8.

```
1 rpm -Uvh http://repo.rundeck.org/latest.rpm
2 yum install rundeck
3 service rundeckd start
```

Koodiesimerkki 8: Rundeckin asennus ja käynnistys

5.1 Päivityssuunnitelma ja toteuttaminen

Kun virtuaalipalvelimet oli konfiguroitu halutunlaisesti, testattiin konfiguroinnin toimivuutta. Erityisesti haluttiin tietää, toimiiko Procountor-sovellus klusteroidussa ympäristössä vai esiintyykö käyttöönotossa ongelmia. Rundeckiin haluttiin konfiguroida esimerkki versiopäivityksestä jatkokehitystä varten. Windows-palvelinten WinRM-ominaisuuden testaaminen jätettiin työstä pois, koska tarkoitus oli keskittyä aluksi vain Linux-palvelinten keskitettyyn hallintaan ja päivitykseen.

5.2 Tomcat

Kun palvelimet oli konfiguroitu halutunlaisesti testattiin sessionjakoa isäntäkoneelta käsin. Kun Tomcat-palvelin käynnistetään, Catalina-komponentti kirjoittaa lokitiedostoa logs/catalina.out-tiedostoon. Catalina.out tiedostoon kirjoitetaan palvelimella toimivien Servlet-sovellusten loki sekä mahdolliset virhetilanteet. Se sisältää myös tietoa palvelimen käynnistystilanteessa sekä siinä ilmenneistä mahdollisista virheistä. Catalinan käynnistystiedostoa muokattiin niin, että Java ottaa käyttöön vain IPv4-protokollapinon, koska laboratorioympäristössä ei ollut konfiguroitu IPv6-protokollaa.

Palvelimen käynnistysvaihteessa havaittiin ongelmia Procountor-sovelluksen sessiotietojen jaossa. Sen käyttämät sessiotiedot eivät kaikki ole serialisoitavia, joka on edellytys sessiotietojen jakamiselle klusterissa. Näin ollen ilman muutoksia sovellukseen ei sessiotietojen jakoa voi ottaa käyttöön. Ongelman korjaaminen vaatisi muutoksia Procountorin lähdekoodiin. Näitä muutoksia ei tässä työssä lähdetty tarkemmin tutkimaan mutta ongelma tullaan kirjaamaan Procountorin kehitykselle tietoon.

```
1 SEVERE: Manager [localhost#/procountor]: Unable to receive
      message through TCP channel
2 java.io.WriteAbortedException: writing aborted; java.io.
      NotSerializableException: java.util.WeakHashMap
```

Koodiesimerkki 9: Java-poikkeus klusterissa

Kun klusterointia ei sellaisenaan voinut ottaa käyttöön, päätettiin palata takaisin Tomcatin oletuskonfiguraatioon ilman klusterointia. Koska aiemmin esitelty AutoDeploy-ominaisuus toimii myös ilman klusteria, voidaan sovelluspäivitys yhä suorittaa ajamatta Tomcatia alas menettämättä sen sisältämiä sessiotietoja. Tämä muuttaa myös Rundeckin suunnitelmaa. Kappaleessa 2.4 mainittu Paraller deployment ei edellytä Tomcatin ajamista klusterissa.

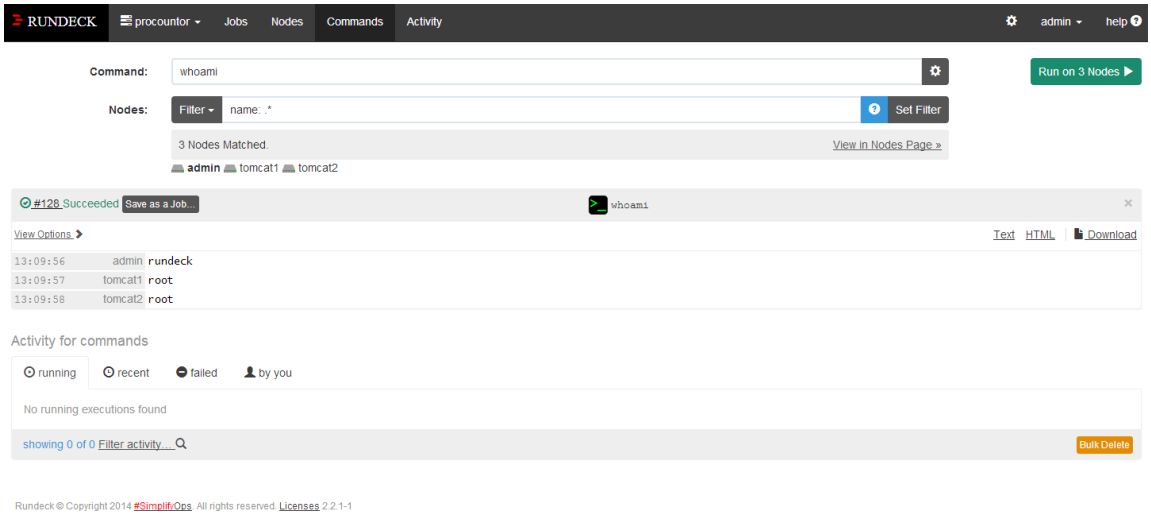
Virtuaaliympäristössä havaittiin myös toinen ongelma, joka aiheutti Tomcatin prosessin jumiutumisen aina, kun Vaadin-sessio vanheni. Tarkempaa syytä prosessin jumiutumiseen ei saatu, mutta se ratkaistiin lopulta vaihtamalla VirtualBox-sovellus VMWare Workstationiin. VirtualBoxista virtuaalikoneet tuotiin VMWareen Export-toimintoa käyttäen.

5.3 Rundeck

Rundeckiä käytetään joko CLI-pohjaisesti tai selaimella. Procountorin käyttöä varten tutustuttiin selaimella toimimaan käyttöliittymään. Tarkempi tutustuminen ohjelmaan aloitettiin tutustumalla Rundeckin ohjekirjaan, jossa käydään läpi sovelluksen toimintaa ja huomioitavia asioita. Tässä työssä käydään esimerkkinä läpi, kuinka Procountorin war-tiedosto voidaan päivittää Tomcat-palvelimille yhtä aikaa ja varmistaa päivityksen eheys.

Runreckin ylin käsitetaso on Projekti. Projektin alle luodaan ryhmiteltäviä Job-elementtejä, jotka suorittavat komentoja eli askelia joko suoraan Rundeck-palvelimella tai erikseen konfiguroiduilla etäpalvelimilla. Etäpalvelinten hallintaan käytetään SSH-protokollaa. Askel voi olla suoritettava mm. komento, suoritettava skripti joko paikallisesti tai tietystä osoitteesta haettuna tai viittaus toiseen Jobiin.

Ennen päivitystä tulee Rundeckiin määritellä sen käyttämät resurssit eli palvelimet mitä Rundeckillä hallitaan. Tarkempi kuvaus resources.xml-tiedostosta kommentteineen on esitelty liitteessä 3. Lisäksi tulee laatia projekti, jonka alle sijoitetaan sijoitetaan Job-tehtävät. Kun resurssit ja projektit oli määritelty, testattiin Commands-välilehdellä SSH-yhteyksien toimivuutta whoami-komennolla. whoami-komento palauttaa Linux-järjestelmässä käyttäjänimen millä komento ajettiin. Kuvankaappaus on esitetty kuvassa 10.

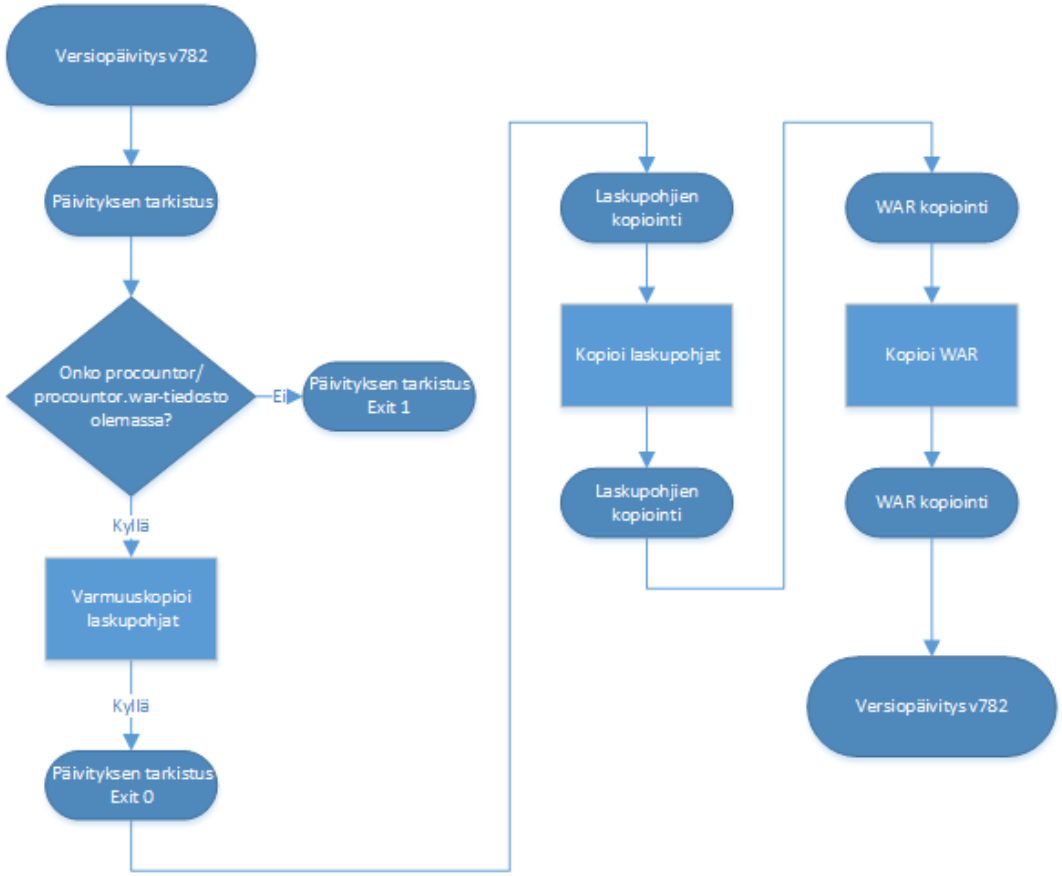


Kuva 10: whoami-komento ajettuna Rundeck-palvelimelta

Jotta komento olisi voitu ajaa palvelimilla, tuli admin-palvelimen julkinen SSH-avain kopioida tomcat-palvelimille. Tämä tapahtui ajamalla admin-palvelimelta komento `ssh-copy-id`.

5.3.1 Procountorin päivittäminen Rundeckillä

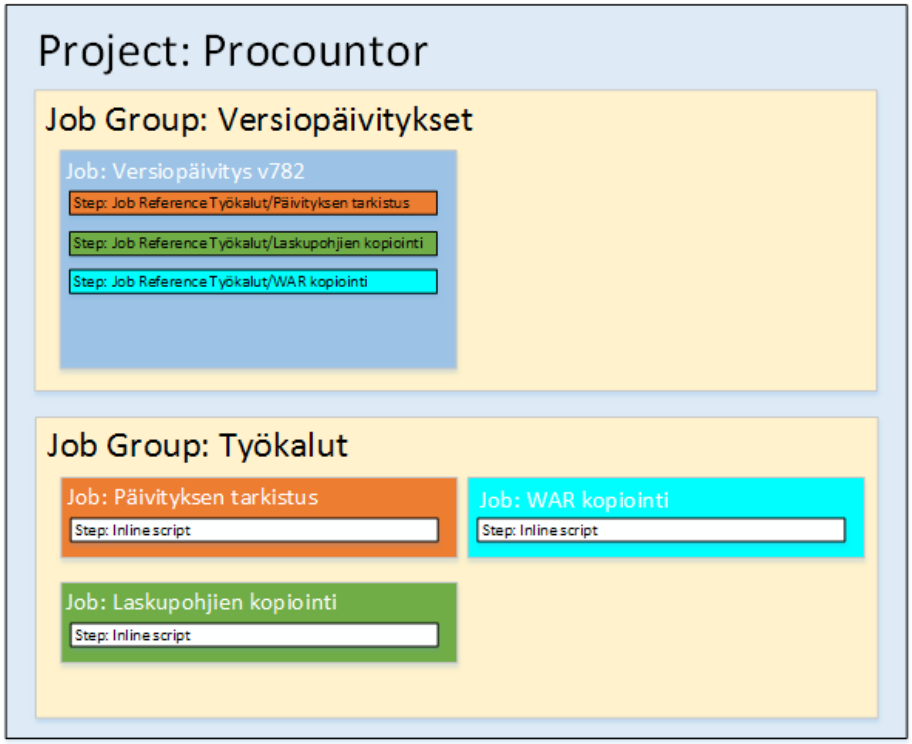
Rundeckiin luotiin aluksi ryhmät `Työkalut` ja `Versiopäivitykset` valmiiksi. Procountorin päivitystä varten laadittiin vuokaavio yksinkertaisesta päivityksen kulusta, joka on esitetty kuvassa 11.



Kuva 11: Procountorin päivityksen vuokaavio

Päivityksessä päivitetään sovelluksen käyttämät laskupohjat sekä procountorin WAR-paketti. Jotta voitaisiin varmistua siitä, että tiedostot ovat varmasti eheitä, kun ne siirretään palvelinten välillä, käytetään rsync-työkalua. Rsync on tiedostojen synkronointiin ja siirtoon tarkoitettu avoimen lähdekoodin ohjelma, joka tarkistaa oletuksena siirrettyjen tiedostojen eheyden MD4-tarkistetta käyttäen. Aiemmin luvussa 2.4 mainittua Tomcatin AutoDeployer-ominaisuutta hyödynnetään niin, että uusi versio otetaan välittömästi

Tomcatin toimesta käyttöön jokaisella palvelimella. Kun suunnittelu vuokaavioineen oli tehty, toteutettiin skriptit, jotka löytyvät liitteestä 3. Skriptit luotiin Rundeckissä `Työkalut`-ryhmän alle. Kuva 12 havainnollistaa, kuinka Job-elementit on aseteltu ryhmiin ja mikä niiden välinen suhde on.



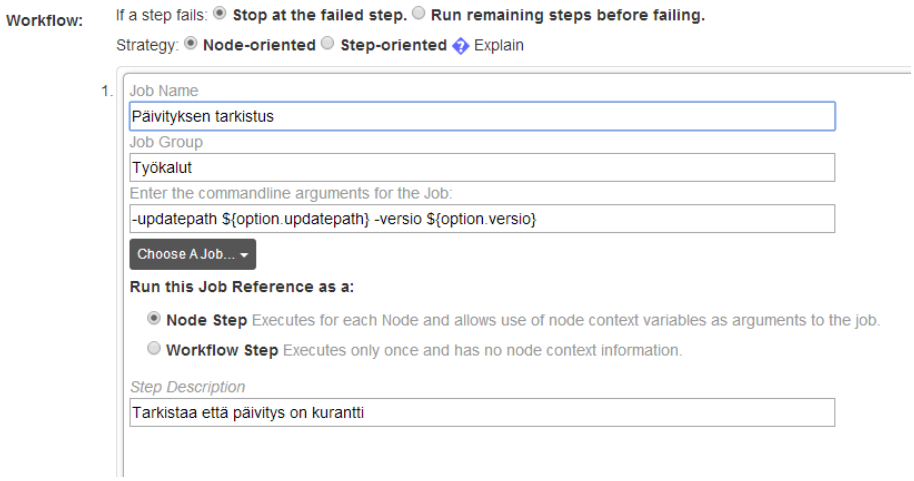
Kuva 12: Projektin ja työkalujen asettelu Rundeckissä

Versiopäivitys v782 päivittää Procountorin Tomcat-palvelimilla uuteen versioon. Siihen luotiin optiot `updatepath` ja `versio`. Nämä optiot annetaan parametreinä päivityksen askeleille. Esimerkki parametreistä on esitetty kuvassa 13 seuraavalla sivulla. Parametrit ovat askelten käytettävissä muuttujissa `${option.updatepath}` ja `${option.versio}`. `updatepath`-parametri kuvaa hakemistoa, missä päivitys on, ja `versio`-parametri kuvaa versiota, mihin Procountor on tarkoitus päivittää.

Päivityksen tarkistus tarkistaa, onko optioilla asetettua hakemistoa olemassa ja sisältääkö se päivitykseen vaadittavat tiedostot. Se myös luo tarvittavat backup-hakemistot. Se suoritetaan admin-palvelimella, joten työlle asetettiin `Node Filter`-kohtaan name: admin.* jotta komentoa suoritettaisiin vain admin-palvelimella.

Laskupohjien kopiointi kopioi laskupohjat admin-palvelimelta yhteiselle levyjaolle, jo-

ka on laboratorioympäristössä samalla palvelimella, käytännössä kopioidaan siis samalle palvelimelle. Komento toimii kuitenkin samalla tavalla myös, jos hakemisto olisi liitetty etäpalvelimeen.



Kuva 13: Job-elementin alla oleville askeleille annetut parametrit

5.3.2 Suoritus

Kun skriptit ja Job-elementit oli saatu luotua, luotiin seuraavaksi `Versiopäivitys v782` mukainen hakemistopolku `/shared/deployment/v7820/` ja sinne edelleen hakemisto invoice_templates ja `procountor/`. Procountor-hakemistoon haettiin taas Procountorin versiokehityksestä version v782 war-paketti ja asetettiin se hakemistoon. Laskupohjat kopioitiin myös kehityspalvelimelta invoice_templates-hakemistoon. Kun hakemistorakenne oli selvillä, voitiin seuraavaksi siirtyä työn suoritukseen.

Versiopäivitys v782 käynnistetään Rundeckissä Jobs-näkymällä olevasta Play-painikkeesta. Työn suoritusta voi tämän jälkeen seurata reaaliajassa Activity-välilehdeltä. Node-palvelinten konsolituloste tulee suoraan Log output-välilehdelle. Mikäli jokin työn askelista aiheuttaa virheen näkyy tämä Report-näkymällä, jossa ilmoitetaan palvelinkohtaisesti askelten onnistumisesta tai epäonnistumisesta. Esimerkki Log Output-välilehden tuottamasta tulosteesta löytyy liitteestä 3.

Päivitys Rundeckillä onnistui nopeasti kaikille palvelimille. Käsityötä ei tarvittu vaan kaikki tapahtui automatisoidusti skriptien perusteella. Jatkokehityksenä Rundeckillä tapah-

tuvaa automatisointia voidaan kehittää edelleen ja sitä laajennetaan koskemaan myös Windows-palvelimia. Kehityskohteita löytyi ainakin seuraavista päivitystoimenpiteistä:

- properties-asetustiedostojen päivitys eli avain=arvo-parien päivitys
- tietokantamuutosten ajo käyttäen Flyway-työkalua
- WinRM-käyttöönotto Windows-palvelimilla
- ylläpitoviestien automaattinen ajastettu asettaminen.

5.4 Yhteenveto

Insinöörityön tarkoituksena oli tutkia, kuinka Procountor-sovelluksen päivitysprosessia voidaan kehittää ja automatisoida, jotta päivityksistä aiheutuisi käyttäjille mahdollisimman vähän haittaa. Tätä lähestyttiin ensin tutkimalla Tomcat-sovelluspalvelimen ominaisuuksia sekä etsimällä palvelinten keskitettyyn hallintaan sopiva avoimen lähdekoodin työkalu. Tarkoitus oli myös tutkia sovelluksen taustalla toimivan tietokantapalvelimen toimintaa ja sen päivittämistä. Painopiste tutkimuksella oli ratkaisujen helpossa ymmärtämisessä ja käyttöönoton sujuvuudessa.

Työn tuloksena saatiin selville, että Procountor ei sellaisenaan ilman muutoksia lähdekoodiin toimi klusteroidussa Tomcat-ympäristössä. Tämä oli tärkeä tieto sovelluksen kehittämisen kannalta, kun käyttäjämäärät kasvavat sovelluksen kansainvälistymisen myötä. Keskitettyyn hallintaan löydettiin erinomainen työkalu Rundeck, johon luotiin esimerkki sovelluksen päivittämisestä ja joukko skriptejä helpottamaan sovelluksen päivittämistä. Myös MySQL-tietokantapalvelimella toimivan tietokannan päivittämistä tarkasteltiin lyhyesti.

Kaiken kaikkiaan tämä insinöörityö palvelee hyvänä katsauksena Procountor-sovelluksen nykytilanteeseen klusteroinnin kannalta ja sisältää seikkoja joita tulee ottaa huomioon sovelluksen päivitysprosessin kehittämisessä edelleen. Löydettiin myös erinomainen työkalu käyttöönotettavaksi Procountorin päivityksiin.

Lähteet

1 Vukotic, Aleksa ja Goodwill, James. Apache Tomcat 7. Yhdysvallat; 2011.

2 Apache Tomcat. wikipedia.org; 2014. Saatavilla: http://en.wikipedia.org/wiki/Apache_Tomcat [Luettu heinäkuu 26, 2014].

3 Clustering/Session Replication HOW-TO. Apache Software Foundation; 2014. Saatavilla: http://tomcat.apache.org/tomcat-7.0-doc/cluster-howto.html [Luettu heinäkuu 20, 2014].

4 MacVittie, Lori Sessions, Sessions Everywhere. 2010;Saatavilla: https://devcentral.f5.com/articles/sessions-sessions-everywhere.

5 The Context Container. Apache Software Foundation; 2014. Saatavilla: http://tomcat.apache.org/tomcat-7.0-doc/config/context.html#Parallel_deployment [Luettu heinäkuu 20, 2014].

6 Karalar, Onur Selecting a Deployment Automation Tool for CRM Software in Elisa Oy. 2013;Saatavilla: http://theseus.fi/bitstream/handle/10024/58164/Automated%20DeploymentV8final.pdf?sequence=1.

7 WS-Management. Wikipedia; 2014. Saatavilla: http://en.wikipedia.org/wiki/WS-Management [Luettu elokuu 16, 2014].

8 Newton, Tim An Introduction to WinRM Basics. 2010;Saatavilla: http://blogs.technet.com/b/askperf/archive/2010/09/24/an-introduction-to-winrm-basics.aspx.

9 Rundeck Introduction. SimplifyOps; 2014. Saatavilla: http://rundeck.org/docs/manual/introduction.html [Luettu elokuu 9, 2014].

10 Rundeck WinRM Plugin. Schueler, Greg; 2014. Saatavilla: https://github.com/rundeck-plugins/rundeck-winrm-plugin [Luettu elokuu 31, 2014].

11 Welcome to Fabric. Forcier, Jeff; 2014. Saatavilla: http://www.fabfile.org/ [Luettu elokuu 9, 2014].

12 What is Capistrano? Capistrano; 2014. Saatavilla: http://capistranorb.com/documentation/overview/what-is-capistrano/ [Luettu heinäkuu 16, 2014].

13 King Tim, Reese George, Yarger Randy, Williams Hugh. Managing and Using MySQL. Yhdysvallat; 2002.

14 Taking the Pain Out of MySQL Schema Changes. Williams, John; 2012. Saatavilla: https://signalvnoise.com/posts/3174-taking-the-pain-out-of-mysql-schema-changes [Luettu elokuu 10, 2014].

15 Schema changes – what's new in MySQL 5.6? Malkowski, Przemysław; 2013. Saatavilla: http://www.mysqlperformanceblog.com/2013/07/05/schema-changes-whats-new-in-mysql-5-6/ [Luettu elokuu 10, 2014].

16 Callaghan, Mark Online Schema Change for MySQL. 2010;Saatavilla: https://www.facebook.com/notes/mysql-at-facebook/online-schema-change-for-mysql/430801045932.

1 MySQL-tietokantapalvelimen Online-operaatiot

Operation	In-Place?	Copies Table?	Allows Concurrent DML?	Allows Concurrent Query?	Notes
CREATE INDEX, ADD INDEX	Yes*	No*	Yes	Yes	Some restrictions for FULLTEXTindex; see next row. Currently, the operation is not in-place (that is, it copies the table) if the same index being created was also dropped by an earlier clause in the sameALTER TABLE statement.
ADD FULLTEXT INDEX	Yes	No*	No	Yes	Creating the first FULLTEXT index for a table involves a table copy, unless there is a user-suppliedFTS_DOC_ID column. SubsequentFULLTEXT indexes on the same table can be created in-place.
DROP INDEX	Yes	No	Yes	Yes	Modifies .frm file only, not the data file.
OPTIMIZE TABLE	Yes	Yes	Yes	Yes	Uses ALGORITHM=INPLACE as of MySQL 5.6.17. ALGORITHM=COPYis used if old_alter_table=1 ormysqld --skip-new option is enabled. OPTIMIZE TABLE using online DDL (ALGORITHM=INPLACE) is not supported for tables with FULLTEXT indexes.
Set default value for a column	Yes	No	Yes	Yes	Modifies .frm file only, not the data file.
Change autoincrementvalue for a column	Yes	No	Yes	Yes	Modifies a value stored in memory, not the data file.
Add a foreign key constraint	Yes*	No*	Yes	Yes	To avoid copying the table, disableforeign_key_checks during constraint creation.

Drop a foreign key constraint	Yes	No	Yes	Yes	The foreign_key_checks option can be enabled or disabled.
Rename a column	Yes*	No*	Yes*	Yes	To allow concurrent DML, keep the same data type and only change the column name.
Add a column	Yes	Yes	Yes*	Yes	Concurrent DML is not allowed when adding an autoincrementcolumn. AlthoughALGORITHM=INPLACE is allowed, the data is reorganized substantially, so it is still an expensive operation.
Drop a column	Yes	Yes	Yes	Yes	Although ALGORITHM=INPLACE is allowed, the data is reorganized substantially, so it is still an expensive operation.
Reorder columns	Yes	Yes	Yes	Yes	Although ALGORITHM=INPLACE is allowed, the data is reorganized substantially, so it is still an expensive operation.
Change ROW_FORMAT property	Yes	Yes	Yes	Yes	Although ALGORITHM=INPLACE is allowed, the data is reorganized substantially, so it is still an expensive operation.
Change KEY_BLOCK_SIZE property	Yes	Yes	Yes	Yes	Although ALGORITHM=INPLACE is allowed, the data is reorganized substantially, so it is still an expensive operation.
Make column NULL	Yes	Yes	Yes	Yes	Although ALGORITHM=INPLACE is allowed, the data is reorganized substantially, so it is still an expensive operation.
Make column NOT NULL	Yes*	Yes	Yes	Yes	When SQL_MODE includesstrict_all_tables orstrict_all_tables, the operation fails if the column contains any nulls. AlthoughALGORITHM=INPLACE is allowed, the data is reorganized substantially, so it is still an expensive operation.
Change data type of column	No	Yes	No	Yes	

Add primary key	Yes*	Yes	Yes	Yes	Although ALGORITHM=INPLACE is allowed, the data is reorganized substantially, so it is still an expensive operation.ALGORITHM=INPLACE is not allowed under certain conditions if columns have to be converted toNOT NULL. See Example 14.9, “Creating and Dropping the Primary Key”.
Drop primary key and add another	Yes	Yes	Yes	Yes	ALGORITHM=INPLACE is only allowed when you add a new primary key in the same ALTER TABLE; the data is reorganized substantially, so it is still an expensive operation.
Drop primary key	No	Yes	No	Yes	Restrictions apply when you drop a primary key primary key without adding a new one in the sameALTER TABLE statement.
Convert character set	No	Yes	No	Yes	Rebuilds the table if the new character encoding is different.
Specify character set	No	Yes	No	Yes	Rebuilds the table if the new character encoding is different.
Rebuild with FORCEoption	Yes	Yes	Yes	Yes	Uses ALGORITHM=INPLACE as of MySQL 5.6.17. ALGORITHM=COPYis used if old_alter_table=1 ormysqld --skip-new option is enabled. Table rebuild using online DDL (ALGORITHM=INPLACE) is not supported for tables with FULLTEXT indexes.
Rebuild with “null” ALTER TABLE ... ENGINE=INNODB	Yes	Yes	Yes	Yes	Uses ALGORITHM=INPLACE as of MySQL 5.6.17. ALGORITHM=COPYis used if old_alter_table=1 ormysqld --skip-new option is enabled. Table rebuild using online DDL (ALGORITHM=INPLACE) is not supported for tables with FULLTEXT indexes.
Set table-level persistent statistics options (STATS_PERSISTENT,STATS_AUTO_RECALCSTATS_SAMPLE_PAGES)	Yes	No	Yes	Yes	Modifies .frm file only, not the data file.

2 Tomcat-palvelinten konfiguraatiot

2.1 tomcat-users.xml

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="manager"/>
  <user username="manager" password="manager" roles="manager
     "/>
</tomcat-users>
```

Koodiesimerkki 10: Tomcat-palvelimen tomcat-users.xml

2.2 server.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!--
  Licensed to the Apache Software Foundation (ASF) under one
      or more
  contributor license agreements.  See the NOTICE file
     distributed with
  this work for additional information regarding copyright
     ownership.
  The ASF licenses this file to You under the Apache License
     , Version 2.0
  (the "License"); you may not use this file except in
     compliance with
  the License.  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing,
      software
  distributed under the License is distributed on an "AS IS"
      BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
     express or implied.
  See the License for the specific language governing
     permissions and
  limitations under the License.
-->
<!-- Note:  A "Server" is not itself a "Container", so you
    may not
     define subcomponents such as "Valves" at this level.
     Documentation at /docs/config/server.html
 -->
```

```
<Server port="8005" shutdown="SHUTDOWN">
  <!-- Security listener. Documentation at /docs/config/
     listeners.html
  <Listener className="org.apache.catalina.security.
     SecurityListener" />
  -->
  <!--APR library loader. Documentation at /docs/apr.html --
     >
  <Listener className="org.apache.catalina.core.
     AprLifecycleListener" SSLEngine="on" />
  <!--Initialize Jasper prior to webapps are loaded.
     Documentation at /docs/jasper-howto.html -->
  <Listener className="org.apache.catalina.core.
     JasperListener" />
  <!-- Prevent memory leaks due to use of particular java/
     javax APIs-->
  <Listener className="org.apache.catalina.core.
     JreMemoryLeakPreventionListener" />
  <Listener className="org.apache.catalina.mbeans.
     GlobalResourcesLifecycleListener" />
  <Listener className="org.apache.catalina.core.
     ThreadLocalLeakPreventionListener" />

  <!-- Global JNDI resources
       Documentation at /docs/jndi-resources-howto.html
  -->
  <GlobalNamingResources>
    <!-- Editable user database that can also be used by
         UserDatabaseRealm to authenticate users
    -->
    <Resource name="UserDatabase" auth="Container"
              type="org.apache.catalina.UserDatabase"
              description="User database that can be updated
                  and saved"
              factory="org.apache.catalina.users.
                 MemoryUserDatabaseFactory"
              pathname="conf/tomcat-users.xml" />
  </GlobalNamingResources>

  <!-- A "Service" is a collection of one or more "
     Connectors" that share
       a single "Container" Note:  A "Service" is not itself
           a "Container",
       so you may not define subcomponents such as "Valves"
          at this level.
       Documentation at /docs/config/service.html
   -->
  <Service name="Catalina">

    <!--The connectors can use a shared executor, you can
```

```
        define one or more named thread pools-->
57      <!--
58      <Executor name="tomcatThreadPool" namePrefix="catalina-
            exec-"
59          maxThreads="150" minSpareThreads="4"/>
60      -->
61
62
63      <!-- A "Connector" represents an endpoint by which
            requests are received
64           and responses are returned. Documentation at :
65           Java HTTP Connector: /docs/config/http.html (
                blocking & non-blocking)
66           Java AJP  Connector: /docs/config/ajp.html
67           APR (HTTP/AJP) Connector: /docs/apr.html
68           Define a non-SSL HTTP/1.1 Connector on port 8080
69      -->
70      <Connector port="8080" protocol="HTTP/1.1"
71                 connectionTimeout="20000"
72                 redirectPort="8443" />
73      <!-- A "Connector" using the shared thread pool-->
74      <!--
75      <Connector executor="tomcatThreadPool"
76                 port="8080" protocol="HTTP/1.1"
77                 connectionTimeout="20000"
78                 redirectPort="8443" />
79      -->
80      <!-- Define a SSL HTTP/1.1 Connector on port 8443
81           This connector uses the BIO implementation that
                requires the JSSE
82           style configuration. When using the APR/native
                implementation, the
83           OpenSSL style configuration is required as
                described in the APR/native
84           documentation -->
85      <!--
86      <Connector port="8443" protocol="org.apache.coyote.
            http11.Http11Protocol"
87                 maxThreads="150" SSLEnabled="true" scheme="
                    https" secure="true"
88                 clientAuth="false" sslProtocol="TLS" />
89      -->
90
91      <!-- Define an AJP 1.3 Connector on port 8009 -->
92      <Connector port="8009" protocol="AJP/1.3" redirectPort="
            8443" />
93
94
95      <!-- An Engine represents the entry point (within
            Catalina) that processes
```

```
96          every request.  The Engine implementation for
               Tomcat stand alone
97          analyzes the HTTP headers included with the request
               , and passes them
98          on to the appropriate Host (virtual host).
99          Documentation at /docs/config/engine.html -->
100
101     <!-- You should set jvmRoute to support load-balancing
           via AJP ie :
102     <Engine name="Catalina" defaultHost="" jvmRoute="jvm1">
103     -->
104     <Engine name="Catalina" defaultHost="localhost">
105
106
107       <Host name="localhost"  appBase="webapps"
108             unpackWARs="true" autoDeploy="true">
109
110
111         <!-- SingleSignOn valve, share authentication
               between web applications
112              Documentation at: /docs/config/valve.html -->
113         <!--
114         <Valve className="org.apache.catalina.authenticator.
               SingleSignOn" />
115         -->
116
117         <!-- Access log processes all example.
118              Documentation at: /docs/config/valve.html
119              Note: The pattern used is equivalent to using
                    pattern="common" -->
120         <Valve className="org.apache.catalina.valves.
               AccessLogValve" directory="logs"
121                prefix="localhost_access_log." suffix=".txt"
122                pattern="%h %l %u %t &amp;quot;%r&amp;quot; %s %b" />
123
124           <!-- Klusterointi -->
125           <Cluster className="org.apache.catalina.ha.tcp.
                 SimpleTcpCluster"
126                channelSendOptions="8">
127
128           <Manager className="org.apache.catalina.ha.session
                 .DeltaManager"
129                    expireSessionsOnShutdown="false"
130                    notifyListenersOnReplication="true"/>
131
132           <Channel className="org.apache.catalina.tribes.
                 group.GroupChannel">
133             <Membership className="org.apache.catalina.
                    tribes.membership.McastService"
134                         address="228.0.0.4"
```

```
135                             port="45564"
136                             mcastBindAddress="tomcat1.
                                   procountor.test"
137                             frequency="500"
138                             dropTime="3000"/>
139                 <Receiver className="org.apache.catalina.
                       tribes.transport.nio.NioReceiver"
140                         address="tomcat1.procountor.test"
141                         port="4000"
142                         autoBind="100"
143                         selectorTimeout="5000"
144                         maxThreads="6"/>
145                 <Sender className="org.apache.catalina.tribes
                       .transport.ReplicationTransmitter">
146                      <Transport className="org.apache.catalina
                           .tribes.transport.nio.
                           PooledParallelSender"/>
147                 </Sender>
148                 <Interceptor className="org.apache.catalina.
                       tribes.group.interceptors.
                       TcpFailureDetector"/>
149                 <Interceptor className="org.apache.catalina.
                       tribes.group.interceptors.
                       MessageDispatch15Interceptor"/>
150     <Interceptor className="org.apache.catalina.tribes.group.
           interceptors.ThroughputInterceptor"/>
151             </Channel>
152             <Valve className="org.apache.catalina.ha.tcp.
                   ReplicationValve"
153                     filter=".*\.gif;.*\.js;.*\.jpg;.*\.png;.*\.
                           css;.*\.txt;"/>
154             <ClusterListener className="org.apache.catalina.ha
                   .session.ClusterSessionListener"/>
155
156             <!-- Farm Deployer -->
157           <Deployer className="org.apache.catalina.ha.deploy.
                 FarmWarDeployer"
158                 tempDir="/services/tomcat/deployment/tempdir/"
159                 deployDir="/services/tomcat/webapps/"
160                 watchDir="/services/tomcat/deployment/watchdir/"
161                 watchEnabled="false"/>
162
163       </Cluster>
164       </Host>
165
166         <!-- Use the LockOutRealm to prevent attempts to guess
                 user passwords
167                 via a brute-force attack -->
168         <Realm className="org.apache.catalina.realm.
               LockOutRealm">
```

```
169          <!-- This Realm uses the UserDatabase configured in
                 the global JNDI
170               resources under the key "UserDatabase".  Any
                     edits
171               that are performed against this UserDatabase
                     are immediately
172               available for use by the Realm.  -->
173          <Realm className="org.apache.catalina.realm.
                 UserDatabaseRealm"
174                 resourceName="UserDatabase"/>
175        </Realm>
176
177         <Host name="tomcat1.procountor.test"  appBase="
               webapps"
178              unpackWARs="true" autoDeploy="true"/>
179
180
181      </Engine>
182    </Service>
183 </Server>
```

Koodiesimerkki 11: Tomcat-palvelimen server.xml

2.3 catalina.sh

```
1 JAVA_OPTS="$JAVA_OPTS -Djava.net.preferIPv4Stack=true"
```

Koodiesimerkki 12: Catalina.sh skriptin muokkaus

2.4 /etc/hosts

```
1 tomcat1.procountor.test 192.168.69.100
2 tomcat2.procountor.test 192.168.69.101
3 admin.procountor.test 192.168.69.200
```

Koodiesimerkki 13: Virtuaaliympäristön IP-osoitteet

2.5 /etc/fstab

```
1
2 # /etc/fstab
3 # Created by anaconda on Thu Jun 26 12:42:14 2014
4 #
5 # Accessible filesystems, by reference, are maintained under
      '/dev/disk'
6 # See man pages fstab(5), findfs(8), mount(8) and/or blkid
     (8) for more info
7 #
8 /dev/mapper/VolGroup-lv_root /                       ext4
        defaults        1 1
9 UUID=bf7d3d9b-dd39-442b-a457-f7768a5b61c3 /boot
                        ext4    defaults        1 2
```

```
10 /dev/mapper/VolGroup-lv_swap swap                    swap
        defaults        0 0
11 tmpfs                   /dev/shm                tmpfs
     defaults        0 0
12 devpts                  /dev/pts                devpts  gid
     =5,mode=620  0 0
13 sysfs                   /sys                    sysfs
     defaults        0 0
14 proc                    /proc                   proc
     defaults        0 0
15 192.168.69.200:/shared  /services/shared_resources/     nfs
         rw,sync,hard,intr       0 0
```

Koodiesimerkki 14: fstab

3 Rundeck-konfiguraatiot ja tulosteet

3.1 Rundeck: resources.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
  <!-- Admin-palvelin -->
  <node name="admin" description="Rundeck server node" tags=
     "" hostname="admin" osArch="amd64" osFamily="unix"
     osName="Linux" osVersion="2.6.32-431.23.3.el6.x86_64"
     username="root"/>
  <!-- Tomcat-palvelimet -->
  <node name="tomcat1" tags="app,production" osFamily="unix"
      hostname="192.168.69.100" username="root" />
  <node name="tomcat2" tags="app,production" osFamily="unix"
      hostname="192.168.69.101" username="root" />
</project>
```

Koodiesimerkki 15: Rundeck-palvelimen resources.xml

3.2 Rundeck: Työkalut/Päivityksen tarkistus

```
#!/bin/bash
# Tarkistaa että päivitys sisältää vaadittavan
   hakemistorakenteen ja tiedostot
# VERSIO: THESIS
echo "[debug] Aloitetaan päivityksen tarkistus"
echo "[debug] Päivityshakemisto: $RD_OPTION_UPDATEPATH"
echo "[debug] Päivityksen versio: $RD_OPTION_VERSIO"

#
# Tarkistetaan onko laskupohjahakemistoa olemassa
#
echo "[debug] Tarkistetaan onko päivityksessä
   laskupohjahakemistoa"
if [ -d $RD_OPTION_UPDATEPATH/$RD_OPTION_VERSIO/
   invoice_templates/ ]; then
  echo "[debug] Laskupohjahakemisto löytyi, luodaan
     varmuuskopiointihakemisto"
  if ! sudo mkdir -p $RD_OPTION_UPDATEPATH/$RD_OPTION_VERSIO
     /backup/invoice_templates; then
    echo "[fail] Laskupohjien varmuuskopiointihakemiston
       luonti epäonnistui"
    exit 1
  fi
else
```

```
19    echo "[debug] Päivitys ei sisällä laskupohjahakemistoa, ei
          voida jatkaa"
20    echo "[debug] Hakemisto on oltava vaikka laskupohjia ei
         olisikaan"
21    exit 1
22  fi
23
24  #
25  # Kopioidaan nykyiset laskupohjat talteen
26  #
27  echo "[debug] Aloitetaan laskupohjien varmuuskopiointi"
28  if ! sudo rsync --checksum -rvi --progress /services/shared/
       invoice_templates $RD_OPTION_UPDATEPATH/$RD_OPTION_VERSIO
       /backup/invoice_templates ; then
29    echo "[fail] Laskupohjien varmuuskopiointi epäonnistui"
30    exit 1
31  fi
32
33  #
34  # Tarkistetaan onko procountor-hakemisto ja sen sisällä war
35  # Varmuuskopiointia ei tehdä koska rollback tapahtuu
       poistamalla uusi versio tuotannosta
36  #
37  echo "[debug] Tarkistetaan onko procountor-hakemisto
       olemassa"
38  if [ -d $RD_OPTION_UPDATEPATH/$RD_OPTION_VERSIO/procountor/
       ]; then
39    echo "[debug] Procountor-hakemisto löytyi, tarkistetaan
         tiedostot"
40    if [ -f $RD_OPTION_UPDATEPATH/$RD_OPTION_VERSIO/procountor
         /procountor.war ]; then
41      echo "[debug] procountor.war OK"
42    else
43      echo "[fail] procountor.war NOT FOUND"
44      exit 1
45    fi
46  else
47    echo "[debug] Procountor-hakemistoa ei löytynyt"
48    exit 1
49  fi
50  echo "[debug] Päivitys on OK!"
```

Koodiesimerkki 16: Rundeck-palvelimen resources.xml

3.3 Rundeck: Työkalut/Laskupohjien kopiointi

```
1  #!/bin/bash
2  # Laskupohjien kopiointi
3  # Versio: THESIS
4
5  echo "[debug] Kopioidaan laskupohjat"
```

```
6
7  if ! rsync --checksum -rvi --progress $RD_OPTION_UPDATEPATH/
       $RD_OPTION_VERSIO/invoice_templates/ /services/shared/
       invoice_templates/ ; then
8    echo "[fail] Laskupohjien kopiointi epäonnistui"
9    exit 1
10 fi
```

Koodiesimerkki 17: Laskupohjien kopiointi

3.4 Rundeck: Työkalut/WAR-kopiointi

```
1
2  #!/bin/bash
3  # Vaadin-päivitys
4  # Versio: THESIS
5  echo "[debug] Siirretään WAR tuotantoon"
6
7  VERSION=$1
8
9  if ! rsync --checksum -rvi --progress /services/
       shared_resources/deployment/$VERSION/procountor/
       procountor.war /services/tomcat/webapps/procountor##
       $VERSION.war ; then
10   echo "[fail] WAR-kopiointi epäonnistui"
11   exit 1
12 fi
```

Koodiesimerkki 18: WAR-kopiointi

```
1
2  #!/bin/bash
3  # Vaadin-päivitys
4  # Versio: THESIS
5  echo "[debug] Siirretään WAR tuotantoon"
6
7  VERSION=$1
8
9  if ! rsync --checksum -rvi --progress /services/
       shared_resources/deployment/$VERSION/procountor/
       procountor.war /services/tomcat/webapps/procountor##
       $VERSION.war ; then
10   echo "[fail] WAR-kopiointi epäonnistui"
11   exit 1
12 fi
```

Koodiesimerkki 19: WAR-kopiointi

3.5 Rundeck: Versiopäivitys v782 suorituksen tuloste

```
1  19:47:51 [root@admin 1@node=tomcat1/1][NORMAL] [debug]
       Aloitetaan päivityksen tarkistus
2  19:47:51 [root@admin 1@node=tomcat1/1][NORMAL] [debug]
       Päivityshakemisto: /services/shared/deployment
3  19:47:51 [root@admin 1@node=tomcat1/1][NORMAL] [debug]
       Päivityksen versio: v7820
```

```
 4 19:47:51 [root@admin 1@node=tomcat1/1][NORMAL] [debug]
     Tarkistetaan onko päivityksessä laskupohjahakemistoa
 5 19:47:51 [root@admin 1@node=tomcat1/1][NORMAL] [debug]
     Laskupohjahakemisto löytyi, luodaan
     varmuuskopiointihakemisto
 6 19:47:51 [root@admin 1@node=tomcat1/1][NORMAL] [debug]
     Aloitetaan laskupohjien varmuuskopiointi
 7 19:47:51 [root@admin 1@node=tomcat1/1][NORMAL] sending
     incremental file list
 8 19:47:52 [root@admin 1@node=tomcat1/1][NORMAL]
 9 19:47:52 [root@admin 1@node=tomcat1/1][NORMAL] sent 15468
     bytes  received 13 bytes  10320.67 bytes/sec
10 19:47:52 [root@admin 1@node=tomcat1/1][NORMAL] total size is
      14913680  speedup is 963.35
11 19:47:52 [root@admin 1@node=tomcat1/1][NORMAL] [debug]
     Tarkistetaan onko procountor-hakemisto olemassa
12 19:47:52 [root@admin 1@node=tomcat1/1][NORMAL] [debug]
     Procountor-hakemisto löytyi, tarkistetaan tiedostot
13 19:47:52 [root@admin 1@node=tomcat1/1][NORMAL] [debug]
     procountor.war OK
14 19:47:52 [root@admin 1@node=tomcat1/1][NORMAL] [debug]
     Päivitys on OK!
15 19:47:52 [root@admin 1@node=tomcat2/1][NORMAL] [debug]
     Aloitetaan päivityksen tarkistus
16 19:47:52 [root@admin 1@node=tomcat2/1][NORMAL] [debug]
     Päivityshakemisto: /services/shared/deployment
17 19:47:52 [root@admin 1@node=tomcat2/1][NORMAL] [debug]
     Päivityksen versio: v7820
18 19:47:52 [root@admin 1@node=tomcat2/1][NORMAL] [debug]
     Tarkistetaan onko päivityksessä laskupohjahakemistoa
19 19:47:52 [root@admin 1@node=tomcat2/1][NORMAL] [debug]
     Laskupohjahakemisto löytyi, luodaan
     varmuuskopiointihakemisto
20 19:47:52 [root@admin 1@node=tomcat2/1][NORMAL] [debug]
     Aloitetaan laskupohjien varmuuskopiointi
21 19:47:52 [root@admin 1@node=tomcat2/1][NORMAL] sending
     incremental file list
22 19:47:52 [root@admin 1@node=tomcat2/1][NORMAL]
23 19:47:52 [root@admin 1@node=tomcat2/1][NORMAL] sent 15468
     bytes  received 13 bytes  30962.00 bytes/sec
24 19:47:52 [root@admin 1@node=tomcat2/1][NORMAL] total size is
      14913680  speedup is 963.35
25 19:47:52 [root@admin 1@node=tomcat2/1][NORMAL] [debug]
     Tarkistetaan onko procountor-hakemisto olemassa
26 19:47:52 [root@admin 1@node=tomcat2/1][NORMAL] [debug]
     Procountor-hakemisto löytyi, tarkistetaan tiedostot
27 19:47:52 [root@admin 1@node=tomcat2/1][NORMAL] [debug]
     procountor.war OK
28 19:47:52 [root@admin 1@node=tomcat2/1][NORMAL] [debug]
     Päivitys on OK!
```

```
19:47:52 [root@admin 2/1][NORMAL] [debug] Kopioidaan laskupohjat
19:47:52 [root@admin 2/1][NORMAL] sending incremental file list
19:47:53 [root@admin 2/1][NORMAL]
19:47:53 [root@admin 2/1][NORMAL] sent 13864 bytes  received 12 bytes  9250.67 bytes/sec
19:47:53 [root@admin 2/1][NORMAL] total size is 14913680 speedup is 1074.78
19:47:55 [root@tomcat1 3/1][NORMAL] [debug] Siirretään WAR tuotantoon
19:47:55 [root@tomcat1 3/1][NORMAL] sending incremental file list
19:47:57 [root@tomcat1 3/1][NORMAL] >f+++++++++ procountor.war
19:47:57 [root@tomcat1 3/1][NORMAL]        32768   0%    0.00kB/s    0:00:00
19:47:58 [root@tomcat1 3/1][NORMAL]     52756480  44%   50.03MB/s    0:00:01
19:47:59 [root@tomcat1 3/1][NORMAL]    104660992  88%   49.67MB/s    0:00:00
19:47:59 [root@tomcat1 3/1][NORMAL]    118133102 100%   49.81MB/s    0:00:02 (xfer#1, to-check=0/1)
19:47:59 [root@tomcat1 3/1][NORMAL]
19:47:59 [root@tomcat1 3/1][NORMAL] sent 118147619 bytes  received 31 bytes  21481390.91 bytes/sec
19:47:59 [root@tomcat1 3/1][NORMAL] total size is 118133102 speedup is 1.00
19:48:02 [root@tomcat2 3/1][NORMAL] [debug] SiirretäänWAR tuotantoon
19:48:02 [root@tomcat2 3/1][NORMAL] sending incremental file list
19:48:05 [root@tomcat2 3/1][NORMAL] >f+++++++++ procountor.war
19:48:05 [root@tomcat2 3/1][NORMAL]        32768   0%    0.00kB/s    0:00:00
19:48:06 [root@tomcat2 3/1][NORMAL]     53018624  44%   49.98MB/s    0:00:01
19:48:07 [root@tomcat2 3/1][NORMAL]    108167168  91%   51.31MB/s    0:00:00
19:48:07 [root@tomcat2 3/1][NORMAL]    118133102 100%   52.68MB/s    0:00:02 (xfer#1, to-check=0/1)
19:48:07 [root@tomcat2 3/1][NORMAL]
19:48:07 [root@tomcat2 3/1][NORMAL] sent 118147619 bytes  received 31 bytes  21481390.91 bytes/sec
19:48:07 [root@tomcat2 3/1][NORMAL] total size is 118133102 speedup is 1.00
```

Koodiesimerkki 20: WAR-kopiointi

3.6 Rundeck järjestelmävaatimukset

The following operating systems are known to support Rundeck:

- Linux: Most recent distributions are likely to work
- Windows: XP, Server and above
- Mac OS X 10.4 or later

Rundeck is a Java-Servlet based server and therefore requires the Java runtime. The install process requires that the latest version of Java 1.7 be installed.